An Intelligent Planning System based on Logical Events

Vasile MAZILESCU *

ABSTRACT

The work reported in this paper serves to promote an Intelligent Planning System that can operate in dynamic and uncertain environments. We can develop and justify thus a series of modelling and design techniques for Intelligent Knowledge Management Systems (IKMS), as well as methods for the analysis of planning systems performance, and, of a fuzzy expert system in particular, between which there are strong similarities. We will also outline a number of differences between conventional problem solving systems and IKMS, the links between expert systems and those of structural and functional planning, the analogy between the model of the problem or process and the problem domain represented by a fuzzy knowledge system based on logical events.

© 2014 EAI. All rights reserved.

1. Introduction

Knowledge-based knowledge management (KBKM) focuses on applications of knowledge-based systems (KBS) tailored to knowledge management (KM) problems. The term appears in (Alavi, 1999) to express the use of KBS to enable knowledge management. KM practitioners and research scientists have been implementing various frameworks to address pragmatic KM problems reusing decades of technology developed for KBS. Therefore, when we talk about knowledge-based knowledge management, we talk about the overlap of knowledge-based systems and knowledge management. It is easy to understand the scope of KM by focusing on a knowledge process that collects, stores and reuses knowledge leveraging it and making organizational the knowledge that once was individual. This knowledge process supports organizational goals by controlling the collection, storage, and use of knowledge. Accordingly, KM applications can be envisioned along the dimensions of a knowledge process that fundamentally performs knowledge tasks to support, steer and control organizational goals. Expert systems (ES), case-based reasoning (CBR), and ontologies are examples of relevant knowledge-based methodologies that have much to contribute to KM systems because they manipulate knowledge to implement various tasks. Although KM practitioners frequently comment that KM is not a technology problem, it is also the case that most KM solutions include an element of technology. It would seem that using technologies for collecting, storing, and reusing knowledge would be critical to consider in any KM effort. Therefore, it is natural that the understanding of a knowledge process differs in different organizations; e.g., a knowledge process in one organization has a different set of tasks than in another. Sometimes these knowledge processes differ on the surface, though their conceptual model is the same. The creation process focuses on how knowledge comes into existence in processes of organizations and individuals. Knowledge understanding comprises verification, represent-action, synthesis, adaptation, storage and organization of the new knowledge with knowledge existing within the organization. For knowledge to be utilized by others in an organization, it must be linked to existing knowledge in the organization in a format that is accepted and understandable by the organization. Knowledge distribution embodies different dissemination methods. Once knowledge is captured and understood, it has to be available for access by individuals in the organization. Knowledge reuse consists of applying knowledge to an organizational process. It is not sufficient to make knowledge available; the reuse step is responsible for promoting reuse by encouraging users to apply available knowledge. This may result in the creation of new knowledge, which in turn begins the knowledge process again (Bhatt, 2000). It is difficult to distinguish a knowledge process from KM and that may be the reason why there are so many definitions on KM as a discipline. We represent a knowledge process as a cycle – one in which KM manifests itself. Can we also say that each of the tasks that are part of a knowledge cycle represent a kind of KM? Our observation is that we can examine a cycle as the knowledge process for KM or we can examine its parts as types of KM. This then gives us a context and framework in which to consider the technologies we will describe in the next sections. We can describe what role they play in the knowledge process and we can also describe how they support the tasks of the knowledge process (another form of KM).

Expert systems represent one way that expertise can be captured, coded, and reused. They are embedded in IKMS and are based on Artificial Intelligence techniques. Fundamentally, an expert system

---

* Faculty of Economics and Business Administrations, Dunarea de Jos University of Galati, Romania. E-mail address: vasile.mazilescu@ugal.ro (V. Mazilescu).
consists of some representation of expertise, some representation of a problem to be solved, and some mechanism to apply the expertise to a problem. Although expertise may be represented in various forms, one common representation for expertise is in the form of rules. The representation of an expert's knowledge would be a collection of rules that were derived from the expert. A rule consists of two parts: an antecedent and a consequent. The rule antecedent consists of one or more conditions that specify when and where to apply the rule. If the conditions of the rule are met, then the second part of a rule – the consequent – specifies the actions to take when the conditions of the rule are met. The mechanism that chooses rules to see if they can be used in a problem is called an inference engine. The inference engine checks antecedents of rules, and based on their values performs the actions specified in the consequents of the rules. To maintain the state of a problem being solved, the inference engine uses a special structure to store the state of problem solution. The structure is called short-term memory. Rules are scanned from the knowledge base to determine which apply to the current problem state in short-term memory. When the inference engine identifies a rule, the actions of that rule are carried out, which may result in a change to the problem state in short-term memory. The process repeats itself until it solves the problem, or no condition can be fulfilled or the expert system is explicitly stopped. There are in IKMS conception three large groups of problems that should approach in terms of decision-based applications: human-environment interface, qualitative knowledge modeling and time management. Such applications obviously require dated event operations the life-time of which should be managed by the system, which often works asynchronously with the acquisition and control system. A real-time expert system shell must also represent imprecise, time and temporal data, encode temporal knowledge, and manage temporal/fuzzy reasoning for allocating temporally interdependent tasks to homogeneous or heterogeneous cooperative agents in dynamic large-scale networks. Planning systems based on Artificial Intelligence use specific models for the problem field, called problem representations and logic argumentation models. Current expert systems have many characteristics in common with planning systems (representation of knowledge, heuristic inference strategies), conceived specifically for communicating with the exterior, while conventional expert systems are strongly encapsulated. Planning systems execute actions in a dynamic manner to produce modification in the state of the problem field. The planner monitors the problem field to progressively obtain information useful for decision synthesis. An explicit loop is being crossed between the actions done by the planner, the problem field, the measured outputs and the planner that uses the outputs to decide the control actions, with a view of reaching the goal. Within expert systems, a similar loop exists: the knowledge base represents the problem field, while the inference engine represents the planner. We have developed an IKMS as an agent that works like a planner. It is realized as a fuzzy real-time expert system shell to meet the challenges of the dynamic environment, like Virtual Organizations (VOs) or Hierarchical Coalitions (HCs). VOs provide an effective instrument to integrate a company's operations with those of other enterprises, to work with customers and create a better product or service, to achieve a faster time to market, and to acquire a higher degree of product customization (Burden 2009, Byrne 1993, Chen 2008, Norman et al. 2004).

Section 2 is a comparative analysis of planning capabilities for IKMS based on related works. At higher levels or at the much more abstract levels, every step of the plan has more effects than a step of the plan at a lower level. Section 3 is dedicated for a presentation of our IKMS based on logical events. Section 4 presents our conclusions and future developments for Distributed Knowledge Management Systems.

2. An analysis of planning capabilities for IKMS

A multi-agent planning architecture (MPA) can be an option in the support systems for coalitions because it is a framework for the integration of various technologies into a system able to solve problems which cannot be solved by the independent systems. The MPA is widely used in planning the applications, such as coalitions' operations and is organized based on the concept of cells planning. Each cell has a managing member that performs the planning of the cell from a group of available agents and distributes the tasks among selected planning agents. Sharing the information generated during the planning process will be held by a central plan, and each agent of the cell can access it. The MPA is important for the integration of the different planning solutions. This integration has limits for the development of support systems coalitions, like any other approach that tries to integrate existing instruments and possibilities in a simple framework. The main reason is collaboration between problem solving on different components and the need to be embedded from the beginning into systems. The problem is how to incorporate collaborative requirements into a distribution of processes planning, so that planning the final connection will not be a sum of independent plans (Stadtler and Kilger 2005). Considering that the most important function of the knowledge-based instruments is to support the human user, we must also understand how they interact in the planning the collaboration process. It should be noted that we do not take into account aspects of the interface which could improve the man-agent interaction. The main idea is that we can join up the groups, associated with the development of hierarchical coalitions as support for the agents, by means of a framework in line with an ontology based on constraints and on the appropriate functions. We can argument a MAP framework brings several advantages such as: a well-known environment to represent and build plans; a transparent manner to integrate collaborative concepts that complement the planning skills; opportunities to develop the human-agent mechanisms; support for customizing the intermediaries. Based on this statement, we can have specific problems.
Computer based information systems (IS) in manufacturing planning and control, have since their introduction evolved to cope with the needs of manufacturing firms. Through the manufacturing resource planning (MRPII) systems in the 1980s and on through enterprise resource planning (ERP) systems in the 1990s ending in recent supply chain management (SCM) software such as Planning and Scheduling Systems - PSS (Bergamaschi et al 2006, Bhatt 2001, David et al 2006, Jonsson et al 2007). Are proposed advanced algorithms in computer system to deal with complicated planning problems. Investing a PSS is expensive and time consuming and it is important that the investment generates a profit. For a considerable time there has been much discussion on the value of IS support in planning and controlling material flows and production (Jonsson and Mattsson 2006, Kim et al 2003, Li and Xiao 2006, Lin et al 2007).

An expert system is a type of planner, since it emulates the way in which experts make decisions, in the presence of abstract, qualitative and approximate knowledge, in a limited expertise field. Currently, what is necessary is the formulation of a mathematical theory of intelligent planning systems, which operates in a dynamic environment, with real time properties (Barachini et al 1990). For systems of Artificial Intelligence applied to management, the fundamental concepts of control theory have a special significance. The state of a planner or an expert system describes the situation that defines the problem solving strategy at a given time. In closed loop planning systems, the planner detects the outputs of the problem field and uses them in the decision making process. Closed loop planners undergo the execution, monitoring and re-planning, being able to continue their activities on the basis of contingency plans that may appear due to the disturbances which act on the problem field. Planning systems, or expert systems, may be designed with three hierarchical levels: the execution level, the coordination level, and the control level. The models used at superior levels are more abstract. A planning problem for a system may be defined thus: given an initial state, a desired state and an aggregate of possible actions, let it an aggregate of actions (either partially or totally ordered) be determined, which applied to the initial state leads the system to the desired state. Planning is a difficult problem and may become much simpler if certain restrictions are applied. Different planning problems may be defined by restricting the type of operators, imposing a series of limitations to the number of preconditions and post conditions. For these types of restrictions, the planning problem is polynomial or NP-complete. The computational complexity of planning was investigated. Bylander analyzes the general problem of deciding the existence of a solution for the planning task in the context of the STRIPS system and it is demonstrated that this general problem is PSPACE-complete (Bylander 1994).

A planning system based on models specific to Artificial Intelligence is made of a planner, the problem field, the connections between the two and the exogenous outputs. The planner’s outputs are inputs for the problem field, representing control actions. The outputs of the problem field are inputs for the planner. They are measured by a planner and used to determine evolution in the problem solving process. Furthermore, there are non-measurable exogenous inputs for the problem field (disturbances), which represent the uncertainty associated with it. The measured exogenous input of the planner represents the goal. It is a task of the planner which examines the outputs of the problem field, it compares them to the goal function and determines what actions must be undertaken to reach it. Not all planners are completely autonomous. Some have a user interface, through which the goals may be generated, allowing for certain degrees of intervention of the human element in the planning process. The planner acts on the problem field through inputs, with a goal to solve a specific problem (Stoop and Wiers 1996, Vollmann et al 2005).

The solution for the problem is a sequence of inputs and outputs (possible states), generated with a view to fulfill the purpose. A model for the field of the real problem is thus developed, called the representation of the problem. Planners based on Artificial Intelligence techniques are made from the following important components (Lunardhi and Passino 1995): the plan generator, the plan simulator (uses plans in the shape of heuristic decision rules), the execution model for the selected plan, the situation evaluator (optional). The Artificial Intelligence (AI) techniques used in the design of planning or expert systems are diverse. They refer especially to representation problems (on must take care in selecting the degree of detail for the mathematical structure used or the allowed modelling power, since too much modelling power may prevent the development of certain components of the planner, the verification and validation of the system), the type of approach (dependant or independent of the field), the type of the planner (hierarchical or not, linear or non-linear, reactive, distributed, encompassing a meta-planner etc.), the type of interactions that may appear in the synthesis of decisions, the forms of searching and replanning.

Due to the strong similarities between the process and the problem field, an analogy may be outlined between the models used for the process and the problem field, on the one side, and between the fundamental systemic concepts on the other. The process is described through stochastic (possibly non-linear) or differential equations, called state and output equations, which describe the dynamics of the process, the structure and its inter-connections. The problem field may be described through symbolic equations. There are still strong mathematical analogies between the two types of systems, studying certain proprieties of special significance in the classic control theory: controllability, observability, stability, system rate. There is a structural analogy between classic control systems and Artificial Intelligence systems (planning, expert system), both in an open loop as well as a closed loop. Generating a plan is the process of synthesis of an aggregate of candidate plans, to fulfil the g, purpose at moment i (the goals can remain fixed or be modified). During the process of plan generation, the system designs (through a simulation based on a model of the problem field) the plausible future plan.
The system uses heuristic plans based on decision rules, resource management, success probabilities, in order to choose the plan to execute. The plan execution module translates the chosen plan into actions over the problem field, using techniques of resource allocation. Situation evaluator uses the inputs, outputs and the representation of the problem in order to determine the state of the field. The estimate state of the field is used for updated the model implicated in the design and generation process of the plan. The term of situation is preferred in this case to designate an abstract, general view of the state. The monitoring of the execution uses the estimated state of the problem field, in order to determine the viable nature of the elaborated plan, in cases of failure launching re-planning actions. In the structure of a planner based on AI techniques may also be integrated a world model, which does the permanent updating of the representation of the problem field (Luger and Stubblefield 1993, McKay and Wiers, 2003).

The evaluation of the situation and the monitoring of the execution in open loop systems cannot be assured, being no possibility of re-planning. In this case, the planner is sensible to the variations which appear in the problem field, open-loop planners not being able to reduce this sensibility. These planners are of interest only for insignificant disturbances, which is not the case with real problems, which need complex and detailed problems. Seeing as the outputs of the system are not detected with inputs, if the problem field is unstable (input-output or internal), then it will never be able to be stabilized in open-loop planning. Open-loop planning absolutely imposes an exact amount of knowledge of the problem field. Since this cannot be obtained, any disturbance may be catastrophic. Open-loop planners present advantages due to their simplicity. If the problem field is stable and the disturbances are insignificant, then these planners may be useful. They also have the advantage of reduced costs, not being necessary measurements of states and outputs related to the problem field. Closed loop planning systems (figure 1) are analogous to closed loop conventional control systems and most of the time, do not use situation evaluation. The monitoring of the execution as well as the re-planning are thus permitted. Our planning system examines the difference between the current output situation and the desired goal, with a view to executing certain actions. The error is not so easily obtained, compared to the classics systems of control, seeing as the distance between the fuzzy states is much harder to quantify. Ever more difficult is the evaluation of the similarities between states, based on a given criteria in the case of our fuzzy expert system. These aspects will clearly come into the design of the IKMS. Designing a IKMS as a planning system is a difficult problem, through the variety of the models used (satisfactory from a computational perspective), which must be a reflection of the complexity of the environment in which they operate.

3. The analysis of our IKMS based on logical events

Definition. The control expert system (CES) is a formal system, defined as (Mazilescu, 2012):

$$CES = (X^CES, E^CES, \delta^CES, g^CES, s^CES)$$

where:

- \(X^CES\times X^E\) is the set of states \(x^CES\) of the control expert system
- \(E^CES\) is the set of input events of the control expert system
- \(\delta^CES\) is the set of state transition functions
- \(g^CES\) is the activation function of the CES
- \(s^CES\) is the initial state
- \(E^CES\subseteq E\) is the set of all trajectories (finite or not) of events in closed loop, which can be generated by the control expert system, based on \(g^CES\) and \(\delta^CES\), and \(E^CES\) is the set of all trajectories of
allowed events in closed loop (a subset of trajectories of events that may result, knowing the trajectories of events of the process and of the expert system, connected together).

Thus, $E_u$ and $P_{CES}$ are viewed as some restrictions in the structure of $R_{CES}$. Based on the trajectories of allowed events, we can highlight additional restrictions which are possible sequences of events within the closed loop system. The output events of fuzzy expert system are considered, in this case, logical events. These issues are particularly important for the future possibility of a qualitative analysis of the closed-loop system (admissibility, cyclical behaviour and stability). Conventional knowledge-based systems can ignore the dynamic behaviour of the control expert system caused by user inputs and process outputs. Many expert systems’ evaluation is done either through difficult simulations, or by comparing its behaviour with the one of human experts. The fuzzy control expert system is like a planner, because it can predict a number of states in the evolution of the process.

An expert system (figure 2) should be designed to remove unwanted behaviours of closed-loop system. Initial state $x_{ES}^{CES}$ is necessary both from theoretical and practical considerations. From the theoretical point of view, it is the beginning of a formal system, in which derivation relations will be the inferential processes, and from the practical point of view, it is necessary to define the initial state of the control expert system in order to reduce possible unwanted combinations of states that could unduly complicate the model. If the initial state of the closed loop system is known, the state transitions can be restricted to the acceptable states of the system. Once specified the initial state $x_{ES}^{CES}=(x_0, x_k)$ for the state $x_k=(x_k, x_{ES}^{CES})$ at time $k$, we get, based on the definition of activation function $g_{CES}(x_k)$, the following form:

$$g_{CES}(x_k)=[g_{ES}(x_k) \cap E_S \cap UI] \cup (g(x_k) \cap HC)[\cup (g(x_k) \cap E_a)]$$

where:
- $g_{ES}(x_k) \cap E_S \cap UI = (P(E_u \cup R) \cap \{0\}) \cap E_S \cap UI$ is the set of input and internal events of the expert system, allowed for the state $x_{ES}^{CES}$;
- $\delta_{ES}(x_k) \cap g(x_k) \cap HC = \{P(E_u \cup E_d) \cap HC \}$ is the set of command input events of the process, allowed for the state $x_{ES}^{CES}$;
- $g(x_k) \cap E_a=(P(E_u \cup E_d) \cap \{0\}) \cap E_a$ is the set of input disturbance events of the process, allowed in the current state $x_k$ of the process.

The input events allowed in the process for the control expert system in closed-loop are the events allowed both by the current state of the process $x_k$ or by the states $x^{CES}_{ES}$, and the events caused by input disturbances of the process, allowed in the state $x_k$. The expert system must control only the activation of the events of type $E_u \cup HC$. It is built in such a way that its transition from a current state in a future state is achieved in response to any output event of the process. Notice that the control expert system’s dynamic must be properly defined, even for the particular case in which the expert system also comprises imprecise knowledge, and its reasoning must reflect the temporal characteristics, specific to process (according to the relationship 2).

![Figure 2. The basic architecture for the CES](image)

Assume that $e_t \in P_{CES}(x_{ES}^{CES})$ is an allowed event of the control expert system in closed loop, currently in the state $x_{ES}^{CES}=(x_0, x_k)$. Under these conditions, if the events $e_t \in g_{ES}(x_k) \cap E_S \cap UI$, $e_a \in \delta_{ES}(x_k) \cap g(x_k) \cap HC$ and $e_d \in g(x_k) \cap E_a$, then $e_t$ can be defined in various ways. These depend on the type of input command events of the process, or on the type of disturbance events, which may occur simultaneously. There is thus a finite number of ways $i_0 \in N^*$, so that for $(\forall) i \in \{1, \ldots, i_0\}$, $e_t^{(i)}$ can be properly defined. Corresponding to each type of event $e_t^{(i)}$, $i=1,\ldots,i_0$, then:

$$e_t^{(i)}(x_k)=x_{ES}^{CES}_{k+1},$$

where $x_{ES}^{CES}_{k+1}=(x_{k+1}, x_{ES}^{CES})$. 


Many AI problems are difficult to solve from the computational point of view. An observation which may help to reduce this complexity is that often these problems have the following property: inputs can be divided into two parts, of which, a part is relatively constant long time, compared with the second part. In such situations, seems right to make some changes in the constant part, in order to reduce the time of obtaining the solution for the AI problem, if the second part varies, but is known at certain moments of time. Transformations made in advance are called pre-processing or knowledge compilation. Using variables in an expert system allows knowledge factorization. First order predicates language facilitates expressing complex knowledge rigorously, imposing appropriate reasoning techniques. Definition of certain propagation and inference procedures for real-time expert systems, involves the development of powerful reasoning mechanisms, as well as adapting the control algorithms to the state spaces, which are often very large.

Our IKMS is conceptually based on all the properties summarized above and consists of the compiled fuzzy rules base (control model) and the inference engine. In order to highlight the characteristics of this system, the following elements must be described: i) The formalism, specifying the types of knowledge supported by the system. Are presented, in order, the fuzzy knowledge syntax and the basic features of the compiled linguistic models, system’s parameters, the elementary fuzzy filtering, the compatibility of possibility distributions for inference scheme; ii) Compiler properties, which include the static structural discrimination component of the fuzzy state $x^{(k)}$, the fuzzy unification tree as the basic element of the structure of compiled fuzzy knowledge (aimed at checking the consistency of fuzzy substitutions), the algorithm that generates the variables linking network. Network’s terminal nodes correspond bijectively to the fuzzy rules; iii) Inference engine algorithm based on fuzzy logic, which includes techniques for reasoning in the presence of compiled imprecise knowledge.

As a first step in the practical implementation of our expert system as an IKMS, knowledge representation aims to describe the problem domain as a model that includes relational entities and symbols, according to an appropriate formalism (Mazilescu, 2012). The types of knowledge accepted by IKMS are: i) variables (symbols always preceded by ‘?’, such as $\mathbf{x}$, $\mathbf{y}$, and which will occur only in rules); ii) atomic constants (numbers or strings); iii) possibility distributions or fuzzy constants (symbols always preceded by the character ‘*’ and used to represent imprecision); iv) logical operators. Possibility distribution can take any form. This complexity can cause a number of difficulties for the application of possibility theory. In practice, when the variable is numeric, it appears that a trapezoidal possibility distribution on continuous referential is well suited. It can be represented through four parameters ($g$, $d$, $\phi$, $\delta$). The trapezoidal form of possibility distributions is preserved in most of the inference and calculation operations. All the fuzzy constants used in knowledge representation and modeling, for the synthesis of fuzzy reasoning algorithms, are represented by trapezoidal possibility distributions, such as $g \leq \delta$, $\phi \leq \delta \geq 0$, called T-numbers. Fuzzy constants can occur both in facts and rules, and are always associated to fuzzy sets (T-numbers) through $Constfaz$ function. Within IKMS, we can equate the fuzzy set to a fuzzy constant. Undefined fuzzy constants are not allowed. A fuzzy constant has always a value corresponding to a continuous, trapezoidal and normalized fuzzy set. Using possibility distributions provides an unified framework for representing imprecision and uncertainty. Parameter $\zeta$ is used to measure fuzzy sets’ uncertainty ($0 \leq \zeta \leq 1$). If a fuzzy set is uncertain, parameter $\zeta$ must be defined in $Constfaz$ function through a list (uncertain $\zeta$). We admit that a completely uncertain fuzzy set ($\zeta=1$) has no effect on system’s behavior. In contrast to facts, a motive is a structured list in which variables may occur. This indicates the presence of variables, atomic constants and of fuzzy constants within motive’s structure. In addition, the motives may occur in both the conditional part and in rules’ conclusion. Uncertainty is allowed in the conditional part and in the consequent of GMP inference scheme, only if a particular linguistic model requires it. In order to increase the knowledge representation capacity, are introduced predicates that appear as motives in the left side of the rules.

**The state space characteristics**

Problem space is characterized by a set of states and operators. An instance of the problem consists of the problem space, an initial state and a set of target states. If several operators may be applied to a particular state and there is not sufficient knowledge to choose between these operators, then is necessary to use the search. In each state selecting the next operator is knowledge retrieving. Application of this knowledge controls the system’s transition from one state to a next state. Search within the problem space is performed at the level of application-specific knowledge. In addition, retrieval knowledge involves searching that available knowledge applicable in the current state, known as the knowledge space. There is a fundamental difference between searching within problem space and searching within the knowledge space. In the space problem, the purpose is to select the best operator which is applicable in the current state, and for the retrieval knowledge, the purpose is to determine all the knowledge that will influence the operator selection. Within problem space, are available knowledge for limiting and controlling the search, but which are not comparable with those for limiting the search, available in the knowledge space. The only way to reduce deviations within the problem space is to reduce the number of searched states, through limiting and abstraction techniques. Deviation reduction for knowledge retrieval can be achieved by partitioning the knowledge space, data partitioning and by reducing knowledge expressiveness. State information contains knowledge that ensures the transition from one state to another, ensures the access information to compiled knowledge base for any of the current states, ensures the return information for system restore in the major ramification points of state space, and the selection of an alternative in case of failure (figure 3).
A state $x^{a}$ of the expert system based on compiled imprecise knowledge has in its structure a set of fuzzy facts (component $x^{b}$). If the antecedent of a rule filters the current state of the expert system, then the actions of this rule can be performed (add or delete facts, procedure call, etc.) in order to obtain a new state of the expert system and hence of the management expert system. Once launched the system, it always has two particular states: initial state and final state.

![Figure 3. The inference engine operation scheme](image)

The system evolves between the two states by chaining the state transitions, like a potential state space tree (search space). Appears, in this case, the risk of combinatorial explosion. For fuzzy expert systems, the problem of evolution within state space is different from the classical case, because of the difficulties in comparing two fuzzy states. Therefore, the system cannot know whether a state is similar to other state, or if it reached the final state, unless there are implemented certain measures to assess the similarity between states. If a state similar to the final state is reached, the engine stops. States comparison is a basic operation for expert systems, but is a very expensive operation. At each inference engine cycle, symbolic comparisons must be performed to determine the engine stop or to avoid cycles (Figure 3). Facts disorder within the states’ structure is a cause for inefficiency in the process of comparing two states. An important advantage of compilation is that it provides a method to characterize the facts that describe system’s states in an orderly manner. Let a fuzzy unification tree with $p$ leaves. After facts propagation within the unification tree, a state $x$ of the management expert system can be represented through a vector with $\alpha$ components of the form, $x = \{x(1), x(2), ..., x(\alpha)\}$, where $x(i)$ is the set of instances of leaf $i$. If $x' = \{x'(1), x'(2), ..., x'(\alpha)\}$ is another state of the system, then $x = x'$ if and only if $x(i) = x'(i)$, $\forall i = 1, ..., \alpha$. If $NRF(x(i))$ is the number of facts from the set of instances $x(i)$, then $x = x'$ if and only if $NRF(x(i)) = NRF(x'(i))$, $\forall i = 1, ..., \alpha$. For $NRF(x(i)) \leq \alpha$, we build a multi-key tree for states comparison. Branch $i$, derived from the root node, corresponds to all previous states for which $NRF(x(i)) = i$. Dispersion method is used to decrease the complexity of states comparison operation. According to our formalism, measures $\Pi(x, x')$ and $N(x, x')$ are used to assess the similarity between states, as follows:

$$
\Pi(x, x') = \min_{i=1, \alpha} \Pi(x(i), x'(i)), N(x, x') = \min_{i=1, \alpha} N(x(i), x'(i))
$$

If $\Pi(x, x') = 1$ and $N(x, x') > 0$, then the state $x$ is similar to state $x$.

At the end of the elementary fuzzy pattern matching stage, if the pattern matching degree satisfies the chosen threshold and if there is a $\sigma$ consistent substitution, then the pattern matching process is a success. The fuzzy condition – fact pattern matching process is the first stage part of the overall behaviour of the inference engine, able to take into consideration the knowledge imprecision. Each instance of a fuzzy reason is associated to a $\sigma$ fuzzy substitution and to the $\Pi, N, \theta, K$ parameters. The second stage in the pattern matching process on a global scale of the fuzzy rules is the fuzzy linking of variables. This conducts the fuzzy unification whose main aim is to verify the consistency of fuzzy substitutions, for which we have already presented a series of theoretical results. Using the tests present in the linking nodes, we can build a dynamic tree that allows adding or suppressing facts. Within each test node of this tree, the values of the variables are tested. If two facts go on the same path, then it is possible that the two facts are consistent. We may use this tree in order to avoid combination challenges. This tree is called linking tree and it is associated to the linking nodes. It presents difficulties for the discrimination of the fuzzy sets within its linking nodes, since certain parts (leaves) of the tree may contain multiple fuzzy facts. That is why the efficiency of this solution decreases, being similar to the use of the unification tree in order to discriminate the fuzzy motives. The main inefficiency factor is related to the disorder of the fuzzy facts in the leaves of the tree. In order to improve the situation, we may use the characteristics of the fuzzy sets in order to sort out facts. This approach was used in order to adapt the unification tree for the processing of fuzzy motives. The major difference that appears between the two situations is the fact that the unification tree is a static tree, i.e. the discriminator motives do
not change, whereas the linking tree is dynamic, the discriminator facts being updated during the functioning period of the inference engine.

The linking process of the fuzzy variables consists of the fuzzy unification and the spreading of the $\Pi$, $N$, $\theta$, $K$ parameters evaluated on a global scale of the antecedent of the rule. We will further insist on the spreading process of the parameters obtained at the end of the pattern matching stage, in the consequence of the rules. The $\Pi$ possibility and $N$ necessity degrees represent the extent to which a rule is satisfied in the current state inside the fact base. During the selection stage, the system selects the rule that satisfies best these conditions in order to trigger it, and the $\theta$ and $K$ parameters serve to apply the Generalised Modus Ponens inference scheme. In the conclusion part of a rule there may be multiple motives (some may be added, others may be deleted, once the rule has been triggered). Before the execution of these actions, we must instantiate all the motives from the right director in which there are three types of data as well as in the conditional director.

The parameters obtained during the pattern matching stage interfere with only two types of data. The linking tree of variables is not a static tree, since we may constantly add or remove facts. The spreading of facts within the linking tree is elementary. We exemplify the adding of the $F \equiv \{s(A \text{ d h}), \sigma(d/?x, h/?y)\}$ fact. This enters at the root and, in the same time tests the value of the $?x$ variable. This value is normally indicated in the substitution associated to $F$ for which $?x = d$. The fact then passes on the branch tagged with a symbol corresponding to the value of the variable in question. In the following node, the value of the $?y$ variable is tested in the same way, and $F$ will pass on the branch tagged with the h symbol. At the end, the $F$ fact is to be found at the level of a leaf within the linking tree and the spreading process thus ends. This moment we must obtain the possible instances, but this does not ensure the fact that these instances are also consistent for the fuzzy case. The instances that satisfy the consistency condition allow the in question rule to be added to the conflicts set. The linking tree is efficient since a fact always follows a unique path. Its depth depends on the number of different tests and its complexity is constant. The linking tree encounters difficulties regarding the discrimination of the fuzzy sets inside its linking nodes, since certain leaves may contain multiple facts.

Take the situation in which the $F_1 = (A \text{ a } b)$, $F_2 = (A \text{ a } c)$, $F_3 = (A \text{ a } d)$, $F_4 = (A \text{ a } e)$, $F_5 = (A \text{ a } g)$, $F_6 = (B \text{ f } h)$ facts occur, for which the present current fuzzy constants correspond to the following fuzzy sets: $(constf\text{az }b\text{tp }10 \ 18 \ 12)$, $(constf\text{az }c\text{tp }7 \ 12 \ 1 \ 1)$, $(constf\text{az }d\text{tp }5 \ 10 \ 1 \ 2)$, $(constf\text{az }e\text{tp }16 \ 20 \ 2 \ 2)$. We have drawn the corresponding linking tree in Figure 5. We see that the leaf that contains the four fuzzy facts is overloaded. Adding the $F_7 = (B \text{ a } e)$ fact by the right input, leads to its memorising in the corresponding field. This fuzzy fact is combined with all the other facts and we must verify the consistency of all combinations. The procedure is obviously inefficient from an algorithmic point of view, the main factor being the disorder of the fuzzy facts.

In order to improve the situation we may use the characteristics of the fuzzy sets. This approach was used in order to adapt the unification tree to the processing of fuzzy motives. The major difference is the fact that the unification tree is a static tree, i.e. the discriminating motives do not change, whereas the linking tree is dynamic. Take $M = \{m_1, \ldots, m_n\}$, a list of fuzzy sets that must be updated, where $m_i = (g_i, \phi_i, \delta_i)$. Assuming that the inferior margins are used, we firstly obtain a $M' = \{m(1), \ldots, m(n)\}$ list. Using the superior margins we
obtain a new list noted \( M' = \{m'(1), \ldots, m'(n)\} \). A new “m fuzzy fact may be rapidly inserted within the \( M' \) or \( M'' \) lists. The problem that we lay down and that must be solved at present is: for a “* fuzzy set defined by the four parameters \( *e = (g_e, d_e, \varphi_e, \delta_e) \), which are the fuzzy sets in \( M \) that are compatible to “*e”. The compatibility between two fuzzy sets has already been analysed, and the estimation of this compatibility depends on the chosen relationship. The compatibility algorithm presents also a series of disadvantages, even though it has the advantage of simplicity and these are: the registration of the two \( M' \) and \( M'' \) lists, if a new fact reaches the level of a leaf it has to undergo twice the algorithm of insertion within the list, and it has to make an intersection operation in order to determine the consistent facts with the new fact.

In order to process the fuzzy sets inside the linking tree, we may use a dynamic discrimination technique of the fuzzy sets (DDMF), based on the I-interval. This tree will be noted DDMF-I and will be taken into account in order to improve the consistency of the fuzzy facts. Everything that follows is based on the \( R \) relation and on the \( \Pi \) possibility measure as a criterion for the compatibility of fuzzy sets.

There are defined \( I(M_s) \) and respectively \( I(M_d) \) minimum intervals for the \( M_s' \) and \( M_d' \) sub-lists. The engendering of the DDMF-I tree is simple and its complexity is not an optimum one. Figure 3 highlights a DDMF-I tree in the fuzzy case, which is more efficient than the tree in Figure 2. This tree will be used for the fuzzy unification. The \( (B \ast e) \) fact will be stored in the right-hand directory, and afterwards we will move on to the searching stage of the consistent instances, by spreading the new fact inside the DDMF-I tree. For the parameters \( \varepsilon=1 \) and \( \eta=0 \) corresponding to the \( R \) relation, we notice that the composition core of the \( R \) relation with “* interCESted with the determined interval after the spreading of all facts inside the DDMF-I tree leads to a non-void result. This means that there may exist facts inside the DDMF-I tree that may be compatible with the new fact. The \( \{ (A \ast e), (B \ast e) \} \), \( \{ (A \ast d), (B \ast e) \} \), \( \{ (A \ast b), (B \ast e) \} \) and \( \{ (A \ast e), (B \ast e) \} \) instances are consistent (Figure 2). In conclusion, the linking nodes represent an important cell in the linking network, and they have a binary and dynamic tree structure, useful for the improvement of the efficiency of the fuzzy inference engine, corresponding to the FRCOM system. From a practical point of view we need to solve the spreading problem of overall parameters that are obtained during the fuzzy condition-fact pattern matching stage and the linking of fuzzy variables over the conclusion of the rule. Firstly, we will take the stock of the parameters obtained during the execution of the pattern matching stage of the control cycle of the fuzzy inference engine. Take a R rule that has \( K \) motives, i.e. \( \text{Cond}(R) = \{ C_1, \ldots, C_K \} \). After the condition-fact fuzzy pattern matching stage, each \( C_i \) condition filters a \( F_i \) fact and we thus obtain a \( \sigma_i \) substitute so that \( F_i = \sigma_i \cdot C_i \) as well as the four parameters \( I(C_i, F_i), N(C_i, F_i), \Theta(C_i, F_i), K(C_i, F_i) \). On the other hand, if there are \( S \) different variables in the conditional part of the R rule, i.e. \( V(\text{Cond}(R)) = \{ \gamma_1, \ldots, \gamma_s \} \), then for each \( \gamma_i \) variable (which may occur several times within the conditional part) we may obtain as a result of the fuzzy substitutions a “*i” fuzzy set as an associated value of the \( \gamma_i \) variable, as well as the \( I(\gamma_i) \) and \( N(\gamma_i) \) measures. The spreading of on the level of the antecedent of the R rule is defined as follows:

\[
\begin{align*}
\Pi &= \min(I(C_i, F_i), I(\gamma_i)) \quad (1 \leq i \leq k, 1 \leq j \leq s) \\
N &= \min(N(C_i, F_i), N(\gamma_i)) \quad (1 \leq i \leq k, 1 \leq j \leq s) \\
\Theta &= \max(\Theta(C_i, F_i)) \quad (1 \leq i \leq k) \\
K &= \min(K(C_i, F_i)) \quad (1 \leq i \leq k)
\end{align*}
\]

For \( \Pi = 1 \) and \( N > 0 \) it is considered that the rule satisfies the pattern matching condition. During the selection stage, the system selects the most appropriate rule to be triggered with the help of these two parameters and also of other selection parameters. \( \Theta \) and \( K \) are used in order to apply the Generalised Modus Ponens inference scheme.

The \textbf{IKMS} inference engine algorithm uses the following variables:

- \( r_{ik} = 1 \), indicates that rule i can be activated at the moment k, and \( r_{ik} = 0 \), means that the rule i cannot be activated at the moment k.
- \( \Pi_{ik}, N_{ik}, \Theta_{ik} \) and \( K_{ik} \) are the values of discrimination parameters existing in the rule's global antecedent, at the time k, in accordance with the intrinsic imprecision of the factual knowledge and of those in the management model based on fuzzy knowledge. These parameters are used in evaluating the difference between rules, their values being updated during the operation of inference engine.
- \( A_{ik} = 1 \) if rule i is deleted from \( MC_k \) or \( A_{ik} = 0 \) if the rule is not deleted from \( MC_k \). This parameter is used to achieve the refraction component of the conflict resolution strategy. Let \( \Lambda = [A_{ik}]^t \) a \( n \times 1 \) -dimensional vector, whose components indicate whether the rule i may be included in the \( MC_k \) if it can be activated for a state \( X_{int} \).
- \( W_{ik} \) is an integer, representing the age of the information at the time k. It thus contributes to pattern-matching the premise of rule i. We define the vector \( W_k = [W_{ik}]^t \) and it can be used in conflict resolution strategy.

In this way, the inference engine state becomes \( X_{int}^{opt} = [A_k W_k \Pi_k N_k \Theta_k K_k]^t \in X_{int} \). Vectors \( A, W, \Pi, N, \Theta \) and \( K \) are implemented in the structure given for the partial state vector \( X_{int}^{opt} \). Completing the model for a management system based on operational knowledge is done by defining the functions \( g^{CES} \) and \( f^{CES} \) (whose values at time k are calculated in relation to the type of event). Imprecise state transitions that occur for updating components \( \Lambda_k \) and \( W_k \) can be based on refraction and on the age of the information in the factual
knowledge base, allowing to update the vector $X^{\text{int}}$. The massive $A$ is used to specify how to update the values of $A_k$ and $W_k$. This is a $n_i \times n_r$ massive, whose components are $a_{k[i][j]} = 1$ if the activation of rule $i$ affects the knowledge pattern-matching for rule $j$, and zero else. This massive contains static information about how the rules from the rule base are interconnected and is specified automatically when the base is loaded, so before the inference process is started. It is a way to model refraction schemes and the age, if they are activated. Refraction component can be done in a special way, using flags of type $F(a)$. In this case, flags setting fulfils the activation sequences of the rules caused by the occurrence of different types of events, the refraction strategy being implemented in the management model. This is the case used in the example developed. Parameters $I_k[i]$, $N_k[i]$, $O_k[i]$ and $K_k[i]$ measure knowledge imprecision and are used in the pattern-matching, selection and propagation processes (inferential process), and their values are variable.

We use variables $V^1_k, V^2_k, V^3_k \ (1 \leq i \leq n_r)$ to define the way to update the state $x_{nk}$ and $W$. $V^i_k$ highlights the difference between the current rule and the other rules of the management model (the higher is this value, the more the rule can be activated). The value of this parameter will be calculated depending on a number of static and dynamic elements of discrimination between rules. Static elements are represented by the number of premises of rules, and those with dynamic nature depend on parameters $N_k[i]$, $O_k[i]$ and $K_k[i]$. Thus, within the inference engine we will use, as elements of discrimination between rules, the actual imprecise value of the knowledge, the initial imprecision of the management model combined with the measured imprecision of the current state. The values of $V^3_k$ indicate the priority of each rule. The value of the static component $V^1_k$ and the value of $V^3_k$ are specified when is defined the knowledge base and remain fixed during the operation of the management model. Their values change, highlighting some sort of subjectivism specific to human decision-maker. The values of $V^1_k$, $W$ and $A$ change with time.

The IKMS Inference Engine Algorithm

**Initializations**: $k = 0, I_k[i] = 0, N_k[i] = 0, O_k[i] = 0, K_k[i] = 0, A_k = 0, W_k = 0, V^2_k = 0, a_{k[i][j]} = 0$.

- **Step 1.** (Identify the current state). Pattern-matching is performed using the compiled fuzzy knowledge structure, specific to IKBSCP system, in relation to the occurrence of a certain type of event at expert system’s input. At the end of this stage we get the set of conflicts $M_{ck}$.

  For $r = 1 \rightarrow n_r$ do

  * If ($r \in M_{ck}$) then $V^3_k = 1$ (Determine the rules that can be activated)

  * If (there is exactly one rule $r'$ so that $F^{\text{int}}(r') = 1$) then execute rule $r'$

  * If (there is not a rule $r'$ so that $F^{\text{int}}(r') = 1$) then stop ($/C = \emptyset$ or call the control module)

- **Step 2.** (Conflict Resolution). This step uses different strategies but which are not all active for a particular fuzzy knowledge model or for a given situation.

- **Step 3.** (Execution)

  $e' = \{r', V^3_k\}$ (we highlight the type of event for the IKMS)

  $X^{\text{int}}_{k[1]} = f^{\text{int}}(X^{\text{int}}_{k[1]}, X^{\text{int}}_{k[2]})$ (as a result of the occurrence of event $e'$, the expert system’s state is updated, i.e. the state component in the facts base and the component $X^{\text{int}}_{k[1]}$, whose update we present below)

  $A_{k[i]}(r') = 1$ (Remove the rules from the set of conflicts, using refraction)

  For $r = 1 \rightarrow n_r$ do

  * If ($r \in M_{ck}$) then $W^3_k = W^3_k + 1$ (Increment the age of each rule from $M_{ck}$)

  * If ($A_{k[i]}(r') = 1$) then $A_{k[i]}(r') = 0$; $W^2_k = 0$ (Allows the rules affected by the activation of rule $r'$ to be considered within the set of conflicts and resets the age of these rules to 0)

End

In the phase of selection based on refraction, if the expert system inference engine, which exploits the management linguistic model developed in accordance with a specific strategy used by certain human decision-maker, cannot respond to the current situation, stop may be replaced by resetting the values of $T^{\text{int}}_{k[i]}$ to the previous values, before applying the refraction and continuing the execution, so that the expert system to use the selection based on refraction in conflict resolution only if this strategy effectively reduces the set of conflicts $M_{ck}$. Note, that $X^{\text{int}}_{k[1]} = f^{\text{int}}(X^{\text{int}}_{k[2]})$, as a result of the occurrence of an event of type $e' = \{r', V^3_k\}$, is the application of the state modifying operator, equivalent to the application of the consequent formula of rule $r'$, selected for execution in the state $X^{\text{int}}_{k[2]}$. In this way are updated also the components of the internal state vector $X^{\text{int}}$ of the inference engine. There was no evidence, within the inference engine algorithm structure, of the mode of action of output function $f^{\text{int}}$, as it has a character specific to the management model. This will be validated practically for the flexible production system, chosen for a case study to highlight the characteristics of our IKMS based on fuzzy knowledge. The closed-loop control expert system can be modelled like a nondeterministic state machine, whose outputs are the process outputs. In this test we emphasize the fact that for a certain initial state of the loads, the balance solution is reached using only the basic rules (R1-R10), i.e. without activating the meta-rules.
The use of temporal aspects refers to the design of those tools to integrate time in control economic applications (e-commerce, Knowledge Management Systems, Virtual Organizations, Multiagent Systems). These aspects are formally found on the inference engine algorithms, able to make full use of the specific knowledge to the process control. We assume that the process operates like finite nondeterministic state.
machine, while the expert system will operate like a finite deterministic state machine. The problem domain must be defined as a collection of problems that the expert system desires to solve. In conventional control, the plant is a dynamical system, described with linear or non-linear differential/ difference equations. A IKMS based on knowledge control and planning models, consists of the planner or the inference engine, the problem domain, the exogenous inputs, and their interconnections. An expert system can be modelled using predicate or temporal logic or other symbolic techniques such as finite state machine. Although the KMSs are frequently being used to perform complex control functions, most often it is the case that no formal analysis of the dynamics is conducted because mathematical analysis of such systems is often considered to be beyond the scope of conventional control theory. The research work reported in this paper serves to promote the development of a firm mathematical foundation on which to perform careful analysis for the verification and validation of KMBSs. There are important another future directions for this work, investigating the dynamics of other reasoning systems that utilize learning and planning in various complex applications, studying computational complexity issues relative to conflict resolution strategies and meta-knowledge representation, and modelling realistic industrial or economic applications that involve Distributed Knowledge Management Systems.

References