KNOWLEDGE ENGINEERING SHELLS pdf

1: Job areas | Shell Global

THEN place tv on wall Y. This rule would take a problem state with an unplaced television and transform it to a state that had the television placed on the opposite wall from the couch. Since the television is now placed, this rule will not fire again. Other rules for other furniture will fire until the furniture arrangement task is finished. Note that for a data driven system, the system must be initially populated with data, in contrast to the goal driven system which gathers data as it needs it. The backward chaining system starts with the goal of finding a value for d and uses the two rules to reduce that to the problem of finding values for a and b. Four levels of data representation Data Representation For all rule based systems, the rules refer to data. The data representation can be simple or complex, depending on the problem. The four levels described in this section are illustrated in figure 1. The most fundamental scheme uses attribute-value pairs as seen in the rules for identifying birds. Examples are color-white, and size-large. When a system is reasoning about multiple objects, it is necessary to include the object as well as the attribute-value. For example the furniture placement system might be dealing with multiple chairs with different attributes, such as size. The data representation in this case must include the object. Once there are objects in the system, they each might have multiple attributes. This leads to a record-based structure where a single data item in working storage contains an object name and all of its associated attribute-value pairs. Frames are a more complex way of storing objects and their attribute-values. Frames add intelligence to the data representation, and allow objects to inherit values from other objects. Furthermore, each of the attributes can have associated with it procedures called demons which are executed when the attribute is asked for, or updated. In a furniture placement system each piece of furniture can inherit default values for length. When the piece is placed, demons are activated which automatically adjust the available space where the item was placed. User Interface The acceptability of an expert system depends to a great extent on the quality of the user interface. The easiest to implement interfaces communicate with the user through a scrolling dialog as illustrated in figure 1. The user can enter commands, and respond to questions. The system responds to commands, and asks questions during the inferencing process. More advanced interfaces make heavy use of pop-up menus, windows, mice, and similar techniques as shown in figure 1. If the machine supports it, graphics can also be a powerful tool for communicating with the user. This is especially true for the development interface which is used by the knowledge engineer in building the system. Scrolling dialog user interface Figure 1. Window and menu user interface Explanations One of the more interesting features of expert systems is their ability to explain themselves. Given that the system knows which rules were used during the inference process, it is possible for the system to provide those rules to the user as a means for explaining the results. This type of explanation can be very dramatic for some systems such as the bird identification system. It could report that it knew the bird was a black footed albatross because it knew it was dark colored and an albatross. It could similarly justify how it knew it was an albatross. At other times, however, the explanations are relatively useless to the user. This is because the rules of an expert system typically represent empirical knowledge, and not a deep understanding of the problem domain. For example a car diagnostic system has rules which relate symptoms to problems, but no rules which describe why those symptoms are related to those problems. Explanations are always of extreme value to the knowledge engineer. They are the program traces for knowledge bases. By looking at explanations the knowledge engineer can see how the system is behaving, and how the rules and data are interacting. This is an invaluable diagnostic tool during development. These include a bird identification system, a car diagnostic system, and a system which places furniture in a living room. Chapters 10 and 11 focus on some actual systems used in commercial environments. These were based on the principles in the book, and use some of the code from the book. It illustrates how to customize a system for a highly specialized problem domain. There is a small semantic gap between Prolog code and the logical specification of a program. This means the description of a section of code, and the code are relatively similar. Because of the small semantic gap, the code examples are shorter and more concise than they might be with another language. The expressiveness of Prolog is due to three major features of the language: The rule-based programming allows the program code to be written in a form which is more declarative than procedural. This is made possible by the built-in pattern matching and backtracking which automatically provide for the flow of control in the program. Together these features make it possible to elegantly implement many types of expert systems. There are also arguments in favor of using conventional languages, such as C, for building expert system shells. Usually these arguments center around issues of portability, performance, and developer experience. As newer versions of commercial Prologs have increased sophistication, portability, and performance, the advantages of C over Prolog decrease. However, there will always be a need for expert system tools in other languages. For those seeking to build systems in other languages, this book is still of value. Since the Prolog code is close to the logical specification of a program, it can be used as the basis for implementation in another language. An in depth understanding of expert systems is not required, but the reader will probably find it useful to explore other texts. In particular since this book focuses on system engineering, readings in knowledge engineering would provide complementary information. Some good books in this area are:

2: Expert Systems and Applied Artificial Intelligence

Offers a systematic approach to knowledge engineering problems. The book gives an overview of knowledge engineering systems and environments, covering classical and recent techniques of their design.

References Modeling Built-in modeling templates, a versatile and user-friendly interface, intuitive controls and features all combine to simplify and expedite a sophisticated object-based modeling process. A broad range of modeling options provide for methods and technologies at the forefront of structural engineering. Model domain may be component, system, or global-level in scope, while encompassing sub-grade components and soil-structure interaction. Grid line, snap, and replication tools are a few of the many practical features which make the modeling environment and process accessible to beginners, and sophisticated for advanced users. Linear or curved members, cables and post-tensioned tendons, link elements to model springs, dampers, isolators, and the associated nonlinear and hysteretic behavior , framing, shell or multi-layered shell, solid elements with isoperimetric formulation and nonlinear response are all modeling options for object assembly in SAP When preferred structural members are not provided in the extensive libraries of SAP, Section Designer is available for custom cross-section design. Users specify geometry and material composition before Section Designer automatically calculates member properties and generates biaxial-interaction and moment-curvature diagrams. Nonlinear-fiber-hinge assignment is another advanced modeling technique available. SAP implements code-based or empirical hinging behavior by modeling geometry and materials as discrete points within a cross-section, then correlating these discretized areas with their associated nonlinear behaviors. Limit-state and hysteretic considerations may also be implemented under nonlinear-static and dynamic analyses. Joint-interpolation algorithms drive automatic edge-constraint technology to connect mesh mismatch. The Reshaper Tool is then available for mesh reshaping and refinement. SAP is the ideal tool for modeling structural systems of any complexity and any project type. Buildings, bridges, transportation infrastructure, such specialty structures as dams, sports facilities, and offshore systems are a few examples of the limitless design possibilities. Loading Powerful built-in templates also simplify and expedite the load-application process. Seismic, wind, vehicle, wave, and thermal forces may all be automatically generated and assigned according to a suite of code-based guidelines. Users are free to define and envelope an unlimited number of load cases and combinations. Moving-load-generation features and a library of AASHTO vehicle applications provide for evaluation of transportation infrastructure systems. For marine systems, wave-load-generation features consider the static and dynamic response of wave, current, buoyancy, and wind while capturing inertial effects. Enveloped load conditions may be coupled with certain advanced analysis and construction techniques P-Delta effect , segmental construction, etc. Analysis A range of innovative analysis techniques are integrated into the capabilities of SAP Users are free to supplement the standard yet sophisticated analysis process by implementing advanced features for nonlinear and dynamic consideration. This versatility makes SAP a practical and productive tool for any analysis type ranging from simple static, linear-elastic to more complex dynamic, nonlinear-inelastic. Options include Eigen analysis with auto shifting for ill-conditioned relations and Ritz analysis for expedited convergence. P-delta effect captures geometric nonlinearity. Buckling analyses provide insight into structural stability through methods characterizing linear buckling which considers multiple buckling modes under nonlinear-static or dynamic application , nonlinear buckling which considers P-delta and large-deflection effects , snap-through buckling, and progressive collapse. Material nonlinearity capture inelastic and limit-state behavior, along with such time-dependent phenomena as creep and shrinkage behavior in reinforced-concrete systems. Plastic hinging may be specified in flexural members according to code-based standards or empirical data. Tension and compression-only springs may be assigned with limits and nonlinear attributes to simulate support plasticity. Static and dynamic methods are available for earthquake simulation. Nonlinear-static-pushover analyses may consider modal , uniform, or user-defined lateral load patterns, plastic-hinging behavior of slender elements, inelastic response of shear walls, floor slabs, and steel plates, and then formulate demand-capacity, damping, and performance-point calculations with customizable summary reports. Dynamic methods include response-spectrum for likely maximum seismic response given pseudo-spectral acceleration vs. Time histories may follow modal or direct-integration methods, and they may be chained together and enveloped with such advanced analyses as P-delta and staged-construction procedures. Staged-construction features are comprehensive. The construction sequence is scheduled with Gantt-chart options, enveloped with performance measures, and paired with analysis procedures. At each construction stage, evaluation may consider static or dynamic structural response, support reactions, geometric and material nonlinearity including buckling, creep, and shrinkage , tendon and cable application with target-tensioning, etc. The Model Alive feature is available for small to medium-sized projects to analyze real or possible structural modifications. Design, Output, and Interoperability Design is fully integrated with the analysis process, enveloping results before automatically sizing steel members and designing reinforced-concrete sections. Automatic steel, concrete, aluminum and cold-formed-framing design code checks ensure that structures meet criteria of American, Canadian, and a variety of international standards. Output and display options are intuitive and practical. Finalized member design, deformed geometry per load combination or mode shape, moment, shear, and axial-force diagrams, section-cut response displays, and animation of time-dependent displacements outline a few of the graphics available upon conclusion of analysis. SAP automatically generates reports for the presentation of images and data. Built-in and customizable templates are available to users for specialized formatting. SAP also provides a suite of interoperability features. Visual Basic and most standard languages are supported. Each subsequent level provides an additional set of features.

3: CiteSeerX â€" Citation Query KEshell: A "Rule Skeleton + Rule Body" Based Knowledge Engineering Sh

Also at Curtis L. Carlson School of Management and Center for Research in Learning, Perception and Cognition, University of Minnesota, Minneapolis, MN Computer Science Department, University of Minnesota, Minneapolis, Minnesota , USA Expert system development methodologies have evolved.

It would match R1 and assert Mortal Socrates into the knowledge base. Backward chaining is a bit less straight forward. In backward chaining the system looks at possible conclusions and works backward to see if they might be true. So if the system was trying to determine if Mortal Socrates is true it would find R1 and query the knowledge base to see if Man Socrates is true. One of the early innovations of expert systems shells was to integrate inference engines with a user interface. This could be especially powerful with backward chaining. So in this example, it could use R1 to ask the user if Socrates was a Man and then use that new information accordingly. The use of rules to explicitly represent knowledge also enabled explanation abilities. In the simple example above if the system had used R1 to assert that Socrates was Mortal and a user wished to understand why Socrates was mortal they could query the system and the system would look back at the rules which fired to cause the assertion and present those rules to the user as an explanation. In English if the user asked "Why is Socrates Mortal? A significant area for research was the generation of explanations from the knowledge base in natural English rather than simply by showing the more formal but less intuitive rules. These systems record the dependencies in a knowledge-base so that when facts are altered, dependent knowledge can be altered accordingly. For example, if the system learns that Socrates is no longer known to be a man it will revoke the assertion that Socrates is mortal. In this, the knowledge base can be divided up into many possible views, a. This allows the inference engine to explore multiple possibilities in parallel. For example, the system may want to explore the consequences of both assertions, what will be true if Socrates is a Man and what will be true if he is not? One of the first extensions of simply using rules to represent knowledge was also to associate a probability with each rule. So, not to assert that Socrates is mortal, but to assert Socrates may be mortal with some probability value. Simple probabilities were extended in some systems with sophisticated mechanisms for uncertain reasoning and combination of probabilities. With the addition of object classes to the knowledge base, a new type of reasoning was possible. Along with reasoning simply about object values, the system could also reason about object structures. In this simple example, Man can represent an object class and R1 can be redefined as a rule that defines the class of all men. These types of special purpose inference engines are termed classifiers. Although they were not highly used in expert systems, classifiers are very powerful for unstructured volatile domains, and are a key technology for the Internet and the emerging Semantic Web. With an expert system the goal was to specify the rules in a format that was intuitive and easily understood, reviewed, and even edited by domain experts rather than IT experts. The benefits of this explicit knowledge representation were rapid development and ease of maintenance. Ease of maintenance is the most obvious benefit. This was achieved in two ways. First, by removing the need to write conventional code, many of the normal problems that can be caused by even small changes to a system could be avoided with expert systems. Essentially, the logical flow of the program at least at the highest level was simply a given for the system, simply invoke the inference engine. This also was a reason for the second benefit: With an expert system shell it was possible to enter a few rules and have a prototype developed in days rather than the months or year typically associated with complex IT projects. A claim for expert system shells that was often made was that they removed the need for trained programmers and that experts could develop systems themselves. In reality, this was seldom if ever true. While the rules for an expert system were more comprehensible than typical computer code, they still had a formal syntax where a misplaced comma or other character could cause havoc as with any other computer language. Also, as expert systems moved from prototypes in the lab to deployment in the business world, issues of integration and maintenance became far more critical. Inevitably demands to integrate with, and take advantage of, large legacy databases and systems arose. To accomplish this, integration required the same skills as any other type of system. Obtaining the time of domain experts for any software application is always difficult, but for expert systems it was especially difficult because the experts were by definition highly valued and in constant demand by the organization. As a result of this problem, a great deal of research in the later years of expert systems was focused on tools for knowledge acquisition, to help automate the process of designing, debugging, and maintaining rules defined by experts. However, when looking at the life-cycle of expert systems in actual use, other problems â€” essentially the same problems as those of any other large system â€” seem at least as critical as knowledge acquisition: This provided a powerful development environment, but with the drawback that it was virtually impossible to match the efficiency of the fastest compiled languages such as C. System and database integration were difficult for early expert systems because the tools were mostly in languages and platforms that were neither familiar to nor welcome in most corporate IT environments â€” programming languages such as Lisp and Prolog, and hardware platforms such as Lisp machines and personal computers. As a result, much effort in the later stages of expert system tool development was focused on integrating with legacy environments such as COBOL and large database systems, and on porting to more standard platforms. These issues were resolved mainly by the client-server paradigm shift, as PCs were gradually accepted in the IT environment as a legitimate platform for serious business system development and as affordable minicomputer servers provided the processing power needed for AI applications. The example applications were not in the original Hayes-Roth table, and some of them arose well afterward. Any application that is not footnoted is described in the Hayes-Roth book.

4: Artificial Intelligence Expert Systems

This paper goes through some of the experiences made when using the expert system shell G2 for one particular application, namely a safety assessment and post-trip guidance system intended for the control room of the Forsmark unit 2 nuclear power plant in Sweden. The experiences made are believed to.

Daniel Bernoulli introduced the principle of virtual work â€" Leonhard Euler developed the theory of buckling of columns Claude-Louis Navier published a treatise on the elastic behaviors of structures Carlo Alberto Castigliano presented his dissertation "Intorno ai sistemi elastici", which contains his theorem for computing displacement as partial derivative of the strain energy. This theorem includes the method of "least work" as a special case Otto Mohr formalized the idea of a statically indeterminate structure. Alexander Hrennikoff solved the discretization of plane elasticity problems using a lattice framework Courant divided a domain into finite subregions Structural failure and List of structural failures and collapses The history of structural engineering contains many collapses and failures. The final collapse killed 94 people, mostly children. In other cases structural failures require careful study, and the results of these inquiries have resulted in improved practices and greater understanding of the science of structural engineering. Some such studies are the result of forensic engineering investigations where the original engineer seems to have done everything in accordance with the state of the profession and acceptable practice yet a failure still eventuated. A famous case of structural knowledge and practice being advanced in this manner can be found in a series of failures involving box girders which collapsed in Australia during the s. Structural engineering theory Figure of a bolt in shear stress. Top figure illustrates single shear, bottom figure illustrates double shear. Structural engineering depends upon a detailed knowledge of applied mechanics , materials science and applied mathematics to understand and predict how structures support and resist self-weight and imposed loads. To apply the knowledge successfully a structural engineer generally requires detailed knowledge of relevant empirical and theoretical design codes , the techniques of structural analysis , as well as some knowledge of the corrosion resistance of the materials and structures, especially when those structures are exposed to the external environment. Such software may also take into consideration environmental loads, such as from earthquakes and winds. Structural engineer Structural engineers are responsible for engineering design and structural analysis. Entry-level structural engineers may design the individual structural elements of a structure, such as the beams and columns of a building. More experienced engineers may be responsible for the structural design and integrity of an entire system, such as a building. Structural engineers often specialize in particular types of structures, such as buildings, bridges, pipelines, industrial, tunnels, vehicles, ships, aircraft and spacecraft. Structural engineers who specialize in buildings often specialize in particular construction materials such as concrete, steel, wood, masonry, alloys and composites, and may focus on particular types of buildings such as offices, schools, hospitals, residential, and so forth. Structural engineering has existed since humans first started to construct their own structures. It became a more defined and formalized profession with the emergence of the architecture as distinct profession from the engineering during the industrial revolution in the late 19th century. Until then, the architect and the structural engineer were usually one and the same thing â€" the master builder. Only with the development of specialized knowledge of structural theories that emerged during the 19th and early 20th centuries, did the professional structural engineers come into existence. The role of a structural engineer today involves a significant understanding of both static and dynamic loading, and the structures that are available to resist them. The complexity of modern structures often requires a great deal of creativity from the engineer in order to ensure the structures support and resist the loads they are subjected to. A structural engineer will typically have a four or five year undergraduate degree, followed by a minimum of three years of professional practice before being considered fully qualified. Structural engineers are licensed or accredited by different learned societies and regulatory bodies around the world for example, the Institution of Structural Engineers in the UK.

KNOWLEDGE ENGINEERING SHELLS pdf

5: Plates and Shells | Mechanical Engineering | MIT OpenCourseWare

This book offers a systematic approach to knowledge engineering problems. It gives a brief overview of knowledge engineering systems and environments, covering both classical and recent techniques of the design and evaluation of them.

It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine. Of course, the term intelligence covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of those. But most progress to date in AI has been made in the area of problem solving -- concepts and methods for building programs that reason about problems rather than calculate a solution. AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called knowledge-based or expert systems. Often, the term expert systems is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often than not, the two terms, expert systems ES and knowledge-based systems KBS , are used synonymously. Taken together, they represent the most widespread type of AI application. The area of human intellectual endeavor to be captured in an expert system is called the task domain. Task refers to some goal-oriented, problem-solving activity. Domain refers to the area within which the task is being performed. Typical tasks are diagnosis, planning, scheduling, configuration and design. An example of a task domain is aircraft crew scheduling, discussed in Chapter 2. Building an expert system is known as knowledge engineering and its practitioners are called knowledge engineers. The knowledge engineer must make sure that the computer has all the knowledge needed to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as symbol patterns in the memory of the computer -- that is, he or she must choose a knowledge representation. He must also ensure that the computer can use the knowledge efficiently by selecting from a handful of reasoning methods. The practice of knowledge engineering is described later. We first describe the components of expert systems. The knowledge base of expert systems contains both factual and heuristic knowledge. Factual knowledge is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field. Heuristic knowledge is the less rigorous, more experiential, more judgmental knowledge of performance. In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the "art of good guessing. One widely used representation is the production rule, or simply rule. The IF part lists a set of conditions in some logical combination. The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving action taken. Expert systems whose knowledge is represented in rule form are called rule-based systems. Another widely used representation, called the unit also known as frame, schema, or list structure is based upon a more passive view of knowledge. The unit is an assemblage of associated symbolic knowledge about an entity to be represented. Typically, a unit consists of a list of properties of the entity and associated values for those properties. Since every task domain consists of many entities that stand in various relations, the properties can also be used to specify relations, and the values of these properties are the names of other units that are linked according to the relations. One unit can also represent knowledge that is a "special case" of another unit, or some units can be "parts of" another unit. The problem-solving model, or paradigm, organizes and controls the steps taken to solve the problem. If the chaining starts from a set of conditions and moves toward some conclusion, the method is called forward chaining. If the conclusion is known for example, a goal to be achieved but the path to that conclusion is not known, then reasoning backwards is called for, and the method is backward chaining. These problem-solving methods are built into program modules called inference engines or inference procedures that manipulate and use knowledge in the knowledge base to form a line of reasoning. The knowledge base an expert uses is what he learned at school, from colleagues, and from years of experience. Presumably the more experience he has, the larger his store of knowledge. Knowledge allows him to interpret the information in his databases to advantage in diagnosis, design, and analysis. Though an expert system consists primarily of a knowledge base and an inference engine, a couple of other features are worth mentioning: Knowledge is almost always incomplete and uncertain. To deal with uncertain knowledge, a rule may have associated with it a confidence factor or a weight. The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called reasoning with uncertainty. An important subclass of methods for reasoning with uncertainty is called "fuzzy logic," and the systems that use them are known as "fuzzy systems. When an answer to a problem is questionable, we tend to want to know the rationale. If the rationale seems plausible, we tend to believe the answer. So it is with expert systems. Most expert systems have the ability to answer questions of the form: The most important ingredient in any expert system is knowledge. The power of expert systems resides in the specific, high-quality knowledge they contain about task domains. AI researchers will continue to explore and add to the current repertoire of knowledge representation and reasoning methods. But in knowledge resides the power. Because of the importance of knowledge in expert systems and because the current knowledge acquisition method is slow and tedious, much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure. Knowledge engineering is the art of designing and building expert systems, and knowledge engineers are its practitioners. Weinberg said of programming in The Psychology of Programming: Knowledge engineering is the same, perhaps more so. We stated earlier that knowledge engineering is an applied part of the science of artificial intelligence which, in turn, is a part of computer science. Theoretically, then, a knowledge engineer is a computer scientist who knows how to design and implement programs that incorporate artificial intelligence techniques. The nature of knowledge engineering is changing, however, and a new breed of knowledge engineers is emerging. Today there are two ways to build an expert system. They can be built from scratch, or built using a piece of development software known as a "tool" or a "shell. Though different styles and methods of knowledge engineering exist, the basic approach is the same: The engineer then translates the knowledge into a computer-usable language, and designs an inference engine, a reasoning structure, that uses the knowledge appropriately. He also determines how to integrate the use of uncertain knowledge in the reasoning process, and what kinds of explanation would be useful to the end user. Next, the inference engine and facilities for representing knowledge and for explaining are programmed, and the domain knowledge is entered into the program piece by piece. It may be that the inference engine is not just right; the form of knowledge representation is awkward for the kind of knowledge needed for the task; and the expert might decide the pieces of knowledge are wrong. All these are discovered and modified as the expert system gradually gains competence. The discovery and cumulation of techniques of machine reasoning and knowledge representation is generally the work of artificial intelligence research. The discovery and cumulation of knowledge of a task domain is the province of domain experts. Domain knowledge consists of both formal, textbook knowledge, and experiential knowledge -- the expertise of the experts. Tools, Shells, and Skeletons Compared to the wide variation in domain knowledge, only a small number of AI methods are known that are useful in expert systems. That is, currently there are only a handful of ways in which to represent knowledge, or to make inferences, or to generate explanations. Thus, systems can be built that contain these useful methods without any domain-specific knowledge. Such systems are known as skeletal systems, shells, or simply AI tools. Building expert systems by using shells offers significant advantages. A system can be built to perform a unique task by entering into a shell all the necessary knowledge about a task domain. The inference engine that applies the knowledge to the task at hand is built into the shell. If the program is not very complicated and if an expert has had some training in the use of a shell, the expert can enter the knowledge himself. Many commercial shells are available today, ranging in size from shells on PCs, to shells on workstations, to shells on large mainframe computers. They range in price from hundreds to tens of thousands of dollars, and range in complexity from simple, forward-chained, rule-based systems requiring two days of training to those so complex that only highly trained knowledge engineers can use them to advantage. They range from general-purpose shells to shells custom-tailored to a class of tasks, such as financial planning or real-time process control. Knowledge acquisition refers to the task of endowing expert systems with knowledge, a task currently performed by knowledge engineers. The power of an expert system lies in its store of knowledge about the task domain -- the more knowledge a system is given, the more competent it becomes. Bricks and Mortar The fundamental working hypothesis of AI is that intelligent behavior can be precisely described as symbol manipulation and can be modeled with the symbol processing capabilities of the computer. In the late s, special programming languages were invented that facilitate symbol manipulation. In the early s another AI programming language was invented in France. Here is an inference rule: A variety of logic-based programming languages have since arisen, and the term prolog has become generic.

6: Home - SAP - Computers and Structures, Inc. - Technical Knowledge Base

Knowledge acquisition refers to the task of endowing expert systems with knowledge, a task currently performed by knowledge engineers. The choice of reasoning method, or a shell, is important, but it isn't as important as the accumulation of high-quality knowledge.

Expert Systems and Applied Artificial Intelligence The field of artificial intelligence AI is concerned with methods of developing systems that display aspects of intelligent behaviour. These systems are designed to imitate the human capabilities of thinking and sensing. Symbolic Processing In AI applications, computers process symbols rather than numbers or letters. AI applications process strings of characters that represent real-world entities or concepts. Symbols can be arranged in structures such as lists, hierarchies, or networks. These structures show how symbols relate to each other. Nonalgorithmic Processing Computer programs outside the AI domain are programmed algorithms; that is, fully specified step-by-step procedures that define a solution to the problem. The actions of a knowledge-based AI system depend to a far greater degree on the situation where it is used. The Field of AI Artificial intelligence is a science and technology based on disciplines such as computer science, biology, psychology, linguistics, mathematics, and engineering. The goal of AI is to develop computers that can think, see, hear, walk, talk and feel. A major thrust of AI is the development of computer functions normally associated with human intelligence, such as reasoning, learning, and problem solving. General problem-solving methods AI established as research field. Knowledge-based expert systems Result: Transaction processing and decision support systems using AI. Resembling the interconnected neuronal structures in the human brain Intelligent agents Result: Software that performs assigned tasks on the users behalf General View The most important applied area of AI is the field of expert systems. An expert system ES is a knowledge-based system that employs knowledge about its application domain and uses an inferencing reason procedure to solve problems that would otherwise require human competence or expertise. It is important to stress to students that expert systems are assistants to decision makers and not substitutes for them. Expert systems do not have human capabilities. They use a knowledge base of a particular domain and bring that knowledge to bear on the facts of the particular situation at hand. The knowledge base of an ES also contains heuristic knowledge - rules of thumb used by human experts who work in the domain. These include areas such as high-risk credit decisions, advertising decision making, and manufacturing decisions. Application areas include classification, diagnosis, monitoring, process control, design, scheduling and planning, and generation of options. An ES is built in a process known as knowledge engineering, during which knowledge about the domain is acquired from human experts and other sources by knowledge engineers. The accumulation of knowledge in knowledge bases, from which conclusions are to be drawn by the inference engine, is the hallmark of an expert system. Knowledge Representation and the Knowledge Base The knowledge base of an ES contains both factual and heuristic knowledge. Knowledge representation is the method used to organize the knowledge in the knowledge base. Knowledge bases must represent notions as actions to be taken under circumstances, causality, time, dependencies, goals, and other higher-level concepts. Several methods of knowledge representation can be drawn upon. Two of these methods include: Frame-based systems - are employed for building very powerful ESs. A frame specifies the attributes of a complex object and frames for various object types have specified relationships. Production rules - are the most common method of knowledge representation used in business. Rule-based expert systems are expert systems in which the knowledge is represented by production rules. A production rule, or simply a rule, consists of an IF part a condition or premise and a THEN part an action or conclusion. The explanation facility explains how the system arrived at the recommendation. Depending on the tool used to implement the expert system, the explanation may be either in a natural language or simply a listing of rule numbers. Inference Engine [Figure Combines the facts of a specific case with the knowledge contained in the knowledge base to come up with a recommendation. In a rule-based expert system, the inference engine controls the order in which production rules are applied Afired and resolves conflicts if more than one rule is applicable at a given time. This is what Areasoning amounts to in rule-based systems. Directs the user interface to query the user for any information it needs for further inferencing. The facts of the given case are entered into the working memory, which acts as a blackboard, accumulating the knowledge about the case at hand. The inference engine repeatedly applies the rules to the working memory, adding new information obtained from the rules conclusions to it, until a goal state is produced or confirmed. Inferencing engines for rule-based systems generally work by either forward or backward chaining of rules. Forward chaining - is a data-driven strategy. The inferencing process moves from the facts of the case to a goal conclusion. The strategy is thus driven by the facts available in the working memory and by the premises that can be satisfied. The inference engine attempts to match the condition IF part of each rule in the knowledge base with the facts currently available in the working memory. If several rules match, a conflict resolution procedure is invoked; for example, the lowest-numbered rule that adds new information to the working memory is fired. The conclusion of the firing rule is added to the working memory. Forward-chaining systems are commonly used to solve more open-ended problems of a design or planning nature, such as, for example, establishing the configuration of a complex product. Backward chaining - the inference engine attempts to match the assumed hypothesized conclusion - the goal or subgoal state - with the conclusion THEN part of the rule. If such a rule is found, its premise becomes the new subgoal. In an ES with few possible goal states, this is a good strategy to pursue. If a hypothesized goal state cannot be supported by the premises, the system will attempt to prove another goal state. Thus, possible conclusions are review until a goal state that can be supported by the premises is encountered. Backward chaining is best suited for applications in which the possible conclusions are limited in number and well defined. Classification or diagnosis type systems, in which each of several possible conclusions can be checked to see if it is supported by the data, are typical applications. Uncertainty and Fuzzy Logic Fuzzy logic is a method of reasoning that resembles human reasoning since it allows for approximate values and inferences and incomplete or ambiguous data fuzzy data. Fuzzy logic is a method of choice for handling uncertainty in some expert systems. Expert systems with fuzzy-logic capabilities thus allow for more flexible and creative handling of problems. These systems are used, for example, to control manufacturing processes. Two important things to keep in mind when selecting ES tools include: The tool selected for the project has to match the capability and sophistication of the projected ES, in particular, the need to integrate it with other subsystems such as databases and other components of a larger information system. The tool also has to match the qualifications of the project team. Expert systems technologies include: Expert system shells - are the most common vehicle for the development of specific ESs. A shell is an expert system without a knowledge base. A shell furnishes the ES developer with the inference engine, user interface, and the explanation and knowledge acquisition facilities. Domain-specific shells are actually incomplete specific expert systems, which require much less effort in order to field an actual system. Expert system development environments - these systems expand the capabilities of shells in various directions. They run on engineering workstations, minicomputers, or mainframes; offer tight integration with large databases; and support the building of large expert systems. ESs are now rarely developed in a programming language. Expert - Successful ES systems depend on the experience and application of knowledge that the people can bring to it during its development. Large systems generally require multiple experts. Knowledge engineer - The knowledge engineer has a dual task. This person should be able to elicit knowledge from the expert, gradually gaining an understanding of an area of expertise. Intelligence, tact, empathy, and proficiency in specific techniques of knowledge acquisition are all required of a knowledge engineer. Knowledge-acquisition techniques include conducting interviews with varying degrees of structure, protocol analysis, observation of experts at work, and analysis of cases. On the other hand, the knowledge engineer must also select a tool appropriate for the project and use it to represent the knowledge with the application of the knowledge acquisition facility. User - A system developed by an end user with a simple shell, is built rather quickly an inexpensively. Larger systems are built in an organized development effort. A prototype-oriented iterative development strategy is commonly used. ESs lends themselves particularly well to prototyping. Problem Identification and Feasibility Analysis: The needed degree of integration with other subsystems and databases is established - concepts that best represent the domain knowledge are worked out - the best way to represent the knowledge and to perform inferencing should be established with sample cases 3. Testing and Refinement of Prototype: End users test the prototypes of the ES. Complete and Field the ES: Benefits and Limitations Expert systems offer both tangible and important intangible benefits to owner companies. These benefits should be weighted against the development and exploitation costs of an ES, which are high for large, organizationally important ESs. But these systems can dramatically reduce the amount of work the individual must do to solve a problem, and they do leave people with the creative and innovative aspects of problem solving. Some of the possible organizational benefits of expert systems are: An Es can complete its part of the tasks much faster than a human expert. The error rate of successful systems is low, sometimes much lower than the human error rate for the same task. ESs make consistent recommendations 4.

7: Expert system - Wikipedia

This paper presents a knowledge engineering shell, KEshell. It adopts a 'rule skeleton + rule body' representation and a linear forward reasoning algorithm. The system structure, the representation scheme, the inference engine and the interactive knowledge acquisition procedure are described.

Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge. The data is collection of facts. The information is organized as data and facts about the task domain. Data, information, and past experience combined together are termed as knowledge. Components of Knowledge Base The knowledge base of an ES is a store of both, factual and heuristic knowledge. Knowledge representation It is the method used to organize and formalize the knowledge in the knowledge base. Knowledge Acquisition The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledge base. The knowledge base is formed by readings from various experts, scholars, and the Knowledge Engineers. The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills. He acquires information from subject expert by recording, interviewing, and observing him at work, etc. The knowledge engineer also monitors the development of the ES. Inference Engine Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution. In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution. Adds new knowledge into the knowledge base if required. Resolves rules conflict when multiple rules are applicable to a particular case. It considers all the facts and rules, and sorts them before concluding to a solution. This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence. It explains how the ES has arrived at a particular recommendation. Verbal narrations in natural language. Listing of rule numbers displayed on the screen. The user interface makes it easy to trace the credibility of the deductions. It should make efficient use of user input. Expert Systems Limitations No technology can offer easy and complete solution. Large systems are costly, require significant development time, and computer resources.

8: Careers | Shell United States

Our TaCIT professionals are working with some of the most innovative IT in the energy industry. Shell's pioneering TaCIT team is one of our most boundary-pushing departments, integrating technical IT with technology and engineering advances.

9: Test Your knowledge on Heat Exchangers â€“ Online Quiz - Chemical Engineering Site

May need advanced knowledge of programming languages and/or development tools in one area, and moderate level skills in multiple areas. Bachelor's degree in Engineering, Computer Science or related technical field, Master's degree may substitute for a portion of the related experience, or equivalent work experience/training.

KNOWLEDGE ENGINEERING SHELLS pdf

Language processing in Chinese Social media as a part of life Identifies for the subject the goals of the ethnographer. The explanation is a Bicycling magazines bicycle touring in the 90s Chapter 2 Creating a poster with Microsoft Word. The monumental and other inscriptions in Halifax Parish Church. Brotherband the invaders Southern Russia and Tunisia Cry of the Invisible The network of fear in your head Mine a practical guide to resource guarding The bears go to town: Grammar Software quality assurance book Bermuda branch of the Jauncey family. Inoculate yourself Kendall ryan hitched espaÃ±ol The castle and other works Machine generated contents note: 1 Conversation in Budapest, 5 Minority Franchise Guide 2008 (Minority Franchise Guide) Come to Christmas One Hundred Years of Phenomenology Captain Careys blunder Principles of Meditation Tooth hypersensitivity Irda annual report 2015-16 in english 30 Days to a More Powerful Memory Current Views on the Prevention, Diagnosis, and Treatment of Hyperlipidaemia The Complete Book of Catholic Colleges, Second Edition (Complete Book of Catholic Colleges) Gujarat samachar epaper vadodara today edition Play like youre dead Shadows in the moonlight Functional programming languages in education 1993 mercedes benz 300se manual Brighouses short forms of wills. Dr bernsteins diabetes solution V. 28]. Arabia, Mesopotamia and Persia The new Europeans Jim Goldberg Freedom of speech in Australian law James, M. R. The haunted dolls house. Elder, A. T. Western panorama: settings and themes in Robert J. C. Stead.