KNOWLEDGE BASED SYSTEMS

Expert System

- An expert system is a computer program that contains stored knowledge and solves problems in a specific field in much the same way that a human expert would.
- The knowledge typically comes from a series of conversations between the developer of the expert system and one or more experts.
- The completed system applies the knowledge to problems specified by a user.

Comparison of Conventional and Expert Systems

Conventional Systems	Expert Systems
Information and its processing are usually combined in one sequential program	Knowledge base is clearly separated from the processing (inference) mechanism (i.e., knowledge rules separated from the control)
Program does not make mistakes (programmers do)	Program .may make mistakes
Do not (usually) explain why input data are needed or how conclusions were drawn	Explanation is a part of most ES
Changes in the program are tedious	Changes in the rules are easy to accomplish
The system operates only when it is completed	The system can operate with only a few rules as fast prototype)
Execution is done on a step-by-step (algorithmic) basis	Execution is done by using heuristics and logic
Need complete information to operate	Can operate with incomplete or uncertain information
Effective manipulation of large databases	Effective manipulation of large knowledge bases
Representation and use of data	Representation and use of knowledge
Efficiency is a major goal	Effectiveness is the major goal
Easily deal with quantitative data	Easily deal with qualitative data
Capture, magnify, and distribute access to numeric data or to information	Capture, magnify, and distribute access to judgment and knowledge

Application Areas of KBS

Area	Problem addressed
Interpretation	Inferring situation descriptions from observations
Prediction	Inferring likely consequences of given situations
Diagnosis	Inferring system malfunctions from observations
Design	Configuring objects under constraints
Planning	Developing plans to achieve goals
Monitoring	Comparing observations to plans, flagging exceptions
Debugging	Prescribing remedies for malfunctions
Repair	Executing a plan to administer a prescribed remedy
Instruction	Diagnosing, debugging, and correcting student performance
Control	Interpreting, predicting, repairing, and monitoring system behaviors

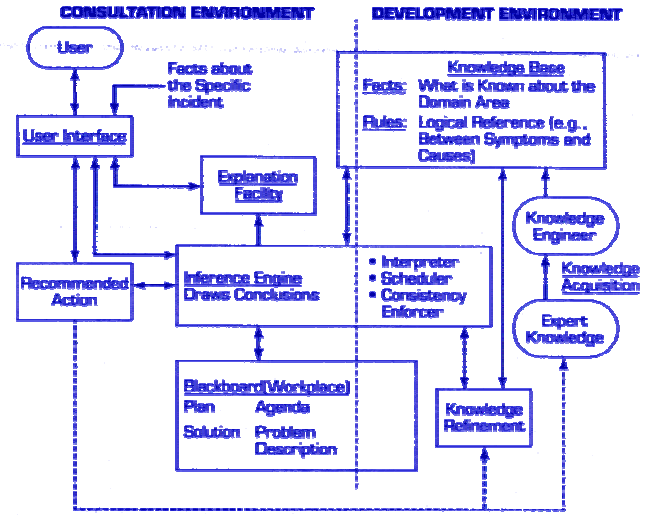
Benefits of KBS

- Increased output and productivity: As compared with humans, KBS can work faster than humans, requiring fewer workers and reducing cost.
- Increased quality: KBS can increase quality by providing consistent advice and reducing error rate.
- Reduced downtime: Using KBS in diagnosing malfunctions and prescribing repairs, it is possible to reduce downtime significantly.
- Capture of scarce expertise
- Flexibility: In providing services and in manufacturing
- Easier equipment operation

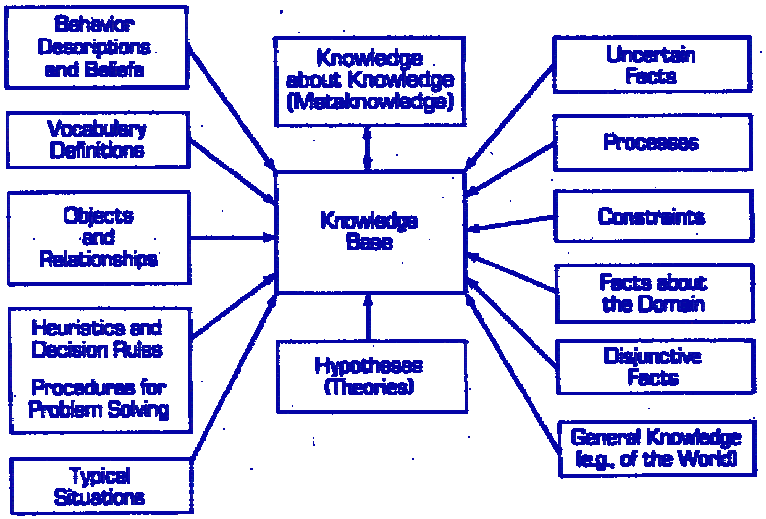
Benefits of KBS

- Elimination of the need for expensive equipment: In many cases a human must rely on expensive instruments for monitoring and control. KBS can perform the same tasks with lower-cost instruments because of their ability to investigate more thoroughly and quickly the information provided by instruments.
- Operation in hazardous environments
- Accessibility to knowledge: KBS make knowledge and information accessible to people.
- Reliability: KBS are reliable in that they do not become tired or bored, and they consistently pay attention to all details and so do not overlook relevant information and potential solutions.
- Increased capabilities of other applications: Integration of KBS with other systems makes the systems more effective; they cover more applications, work faster, and produce higher quality results.
- Ability to work with incomplete and uncertain information

Structure of an Expert System



Knowledge types



Domain Knowledge vs Case Knowledge |

Expert knowledge is mainly expressed by rules like:

IF:

(1) stain of organ is Gram neg. and.
(2) morphology of organ, is rod and
(3) aerobicity of organism is aerobic

THEN:

strong evidence (0.8) that class of organism is Enterobacteriaceae

Case specific knowledge by facts like knowledge about ORGANISM-1:

GRAM =(GRAMNEG 1.0)

MORPH=(ROD 0.9) (COCCUS 0.2)

AIR =(AEROBIC 0.6)

Inferencing and control

- Inferencing is central to reasoning in knowledge-based systems. Assessment of the current situation, producing new knowledge and information to be used in the reasoning process, producing solutions (plans, action decisions, etc.) are all done by employing some sort of inferencing technique.
- Control refers to the control of the execution and communication processes within a knowledge-based system. Control mechanisms supported by a KBS tool depend partly on the paradigm used by the tool.

Requirements of Control Strategies

- Expressiveness
- Adequacy (naturalness)
- Modularity
- Independence of rule base and control strategy
- Sensitivity (i.e., strategy should react on dynamic changes)
- Stability (continuity in behavior)
- Compositionality (possible to add new rules without major unwanted effects)

Inference Rules

Deductive Inference Rule:

Modus ponens: Conclude from "A" and "A implies B" to "B".

A

$\underline{A \Rightarrow B}$

B

Example:

It is raining.

If it is raining, the street is wet.

The street is wet.

Abductive Inference Rule:

Conclude from "B" and "A implies B" to "A".

B

$\underline{A \Rightarrow B}$

A

Example:

The street is wet.

<u>If it is raining, the street is wet.</u>

It is raining.

Rule Invocation

- In rule-based systems rules are invoked typically in three ways:
  - at fixed time intervals (time-triggered)
  - when specified data changes (data-driven, forward-chaining)
  - when needed to achieve a goal (goal-driven, backward-chaining)

Recognize-Act Cycle

- A Rule Interpreter can be described as a recognize-act cycle

1. Match the premise patterns of rules against elements in the working memory
2. If there is more than one rule that can be applied (i.e. that can be "red"), choose one to apply in the conflict resolution. If no rule applicable, stop.
3. Apply the chosen rule, perhaps by adding a new item to the working memory or deleting an old one. If termination condition fulfilled stop, else go to step 1.

- The termination condition is either defined by a goal state or by a cycle condition (e.g. maximally 100 steps)

Matching

In general there are **variables** in the rules, which have to be **matched** when the rule is applied. Variables stand for arbitrary objects.

Consider:

a rule (about the value of horses)

IF:

x is-a horse

x is-a-parent-of y

y is fast

THEN:

x is valuable

and the facts

Comet	is-a	horse
Prancer	is-a	horse
Comet	is-a-parent	Dasher
Comet	is-a-parent	Prancer
Prancer	is	fast
Dasher	is-a-parent	Thunder
Thunder	is	fast
Thunder	is-a	horse
Dasher	is-a	horse

Find bindings for *x* & *y* so that rule is applicable

The bindings for *x* is-a horse:

[x/Comet],

[x/Prancer],

[x/Thunder], and [x/Dasher]

The bindings for *y* is fast

[y/Prancer] and

[y/Thunder]

The bindings for *x* and *y* in *x* is-a-parent-of *y* are:

[x/Comet], [y/Dasher]; [x/Comet], [y/Prancer]; and [x/Dasher], [y/Thunder]

i.e., "Comet is-a horse" matches "*x* is-a horse" but "Comet is-a lion" does not match "*x* is-a horse"

Rule applicable with bindings

[x/Comet], [y/Prancer] and [x/Dasrier], [y/Thunder]

That is new facts that can be derived are:

valuable(Comet) and valuable(Dasher)

- Assume variables x, y, and z and the rule

  IF: x is parent of y

  y is parent of z

  THEN: x is grandparent of z
- and the facts:
  - Lydia is parent of Tony's mother
  - Sam's uncle is parent of Anne
  - Anne's cousin is parent of Lydia
  - Sam is parent of John
  - Anne is parent of Luke
  - John is not parent of Barbara
- Which facts match the preconditions of the rule?
- What can be derived?

Problems with Recognize-Act Cycle

- Efficiency problem with matching:
  - Naive approach of recognize-act cycle.
  - In order to find all applicable rules, try to match all elements in the working memory against all premises in all rules,
  - i.e. for n rules with m premises on average and k elements in the working memory, it is necessary to check n*m*k possible matches in each single cycle.
- Selection of good rule:
  - Different decisions may lead to different outcome.
  - Hence the decision in conflict resolution which rules to apply are crucial.
  - No general solution, but some strategies.

Forward and Backward Chaining

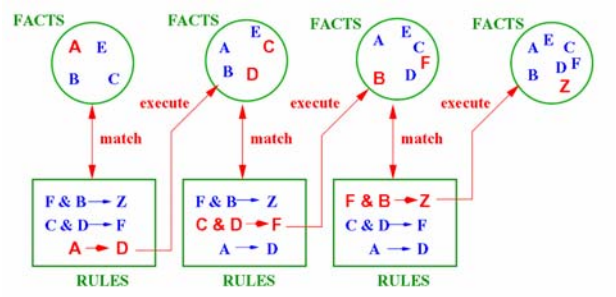
- Expert system shells usually offer one of two reasoning (chaining) modes:
  - data driven or forward chaining; and
  - goal-driven of backward chaining.
- Forward and backward chaining are search techniques used in “if-then” rule systems.
- Which side of the rule is considered first determines the direction of chaining.

Forward Chaining

- In forward chaining, the system begins with known facts about the problem and goes through the rules in the knowledge base trying to assert new facts.
- Rules whose left-hand side (IF part or premise) is known to be true are fired, meaning their right-hand side (THEN part, or conclusion) is declared true.
- This process continues until no more rules can be fired. The system then reports its conclusions.
- Forward-chaining rules are also called antecedent rules.

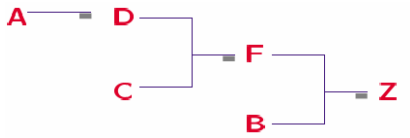
Forward Chaining

- Forward chaining or data-driven inference works from an initial state, and by looking at the premises of the rules (IF-part), perform the actions (THEN-part), possibly updating the knowledge base or working memory.
- This continues until no more rules can be applied or some cycle limit is met, e.g.



Forward Chaining (Cont'd)

- In the example: no more rules, that is, inference chain for this is:



- Problem with forward chaining:
  - many rules may be applicable.
  - The whole process is not directed towards a goal.

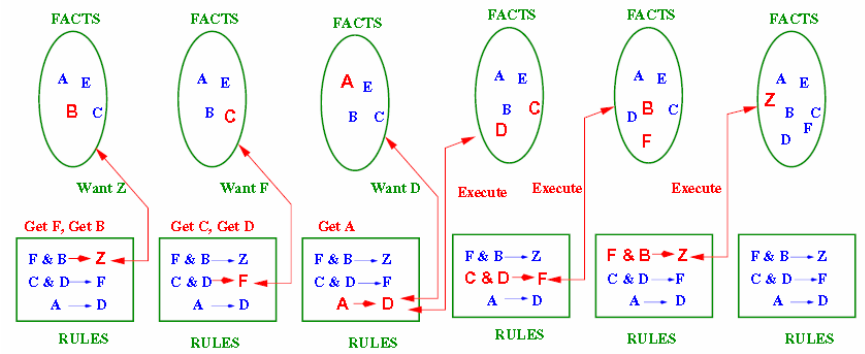
Backward Chaining

- Backward-chaining inference engines start with a goal, or hypothesis, and work through the rules trying to match that goal with the action clauses (THEN part) of a rule.
- When a match is found, the condition clauses (IF part) of the matching rule become a “subgoal” and the cycle is repeated until a verifiable set of condition clauses is found.
- Backward-chaining rules are also called consequent rules.

Backward Chaining

- Backward chaining or goal-driven inference works towards a final state, and by looking at the working memory to see if goal already there.
- If not look at the actions (THEN-parts) of rules that will establish goal, and set up subgoals for achieving premises of the rules (IF-part).
- This continues until some rule can be applied, which is then applied to achieve goal state.
- Advantage of backward chaining:
  - search is directed
- Disadvantage of backward chaining:
  - goal has to be known

Backward Chaining (Cont'd)

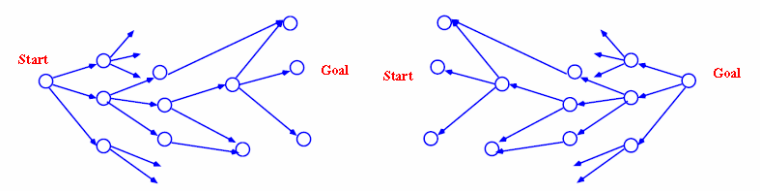


Forward or Backward Reasoning?

- Four major factors
  - More possible start states or goal states?
    - Move from smaller set of states to the larger
    - Consider situation of traveling between home and some unfamiliar place.
      - forward chaining to get home from unfamiliar place.
      - backward chaining to get from home to unfamiliar place.
  - Has program to justify reasoning?
    - Prefer direction that corresponds more closely with the way users will think.
  - What kind of events triggers problem-solving?
    - If it is arrival of a new fact, forward chaining makes sense
    - If it is a query to which a response is required, backward chaining more natural

Forward or Backward Reasoning?

- In which direction is branching factor greatest?
  - Go in the direction with lower branching factor:



backward chaining better forward chaining better

Problems

- Add fact to Raining example from above that person is not dry, hence:
  - Initial facts: State(Dry), Raining, Outside
  - Result of inference now:
  - "State(Dry), Raining, Outside, State(Wet), Develop(Cold)"
  - Inconsistent!
- How to check in case of thousands of rules and facts?
  - By truth maintenance systems
- Which global strategy to apply? Forward or backward chaining?
- Which local strategy to apply, that is, which rule to select in the recognize-act cycle?
- Complexity problem for rule matching, solved by Rete algorithm

The Rete-Algorithm

- Observation:
  - Rules have parts of the conditions in common (called “structural similarity")
  - By the application of a rule the working memory remains almost unchanged (called “temporal redundancy")
- Method:
  - Compile condition parts of the rule in a net (Italian for net: “rete")
  - The net encodes the condition parts (IF-parts) of the rules.
  - The input are the changes of the working memory (i.e. new elements or deleted elements.
  - Modification of elements is simulated by first delete then add modified version)
  - The output is the conflict set (i.e., the applicable rules).

The Rete-Algorithm (Cont.)

- General Procedure:
  - Represent all rules in one net (as above)
  - newly generated working memory elements infiltrate the net from above. If they arrive at the bottom, they enlarge the conflict set with the rule instance, else they are added as internal nodes.
  - newly deleted working memory elements are treated analogously, but in this case the conflict set is possibly narrowed.
- Summary:
  - By sharing and considering updates of the working memory elements only, the efficiency problem can be mainly solved.

Conflict Resolution

- When a search technique completes an evaluation cycle it creates a conflict set made of rules which, during the evaluation cycle, qualified to be fired.
- The set may be empty, in which case the expert system will terminate, or it may contain n number of rules.
- In the latter case, the inference engine uses a conflict resolution strategy to decide exactly which rule should be fired.

Conflict Resolution Strategies

- Rule-Assigned Priority
  - Antecedent Priority,
  - Consequent Priority,
- Complexity,
- Simplicity,
- Specificity,
- Recency,
- Reverse Recency, and
- Random Selection.

Certainty Measurement

- KBS should be able to handle incomplete, inexact, or uncertain knowledge or data. Most of the interesting problems contain uncertainty.
- Uncertainty in expert systems can be traced to the following major sources:
  - information can be unreliable;
  - descriptive (or implementation) languages lack precision;
  - inferences are sometimes drawn with incomplete information;
  - experts sometimes disagree and
  - information can be conflicting.

Paradigms for Dealing with Uncertainty

- Bayes' Theorem,
- Certainty Factor (CF) Model),
- Dempster-Shafer Theory,
- Fuzzy Set Theory,
- Subjective Probability Theory,
- Cohen's Theory of Endorsements, Inheritance, and
- Certainty Thresholds

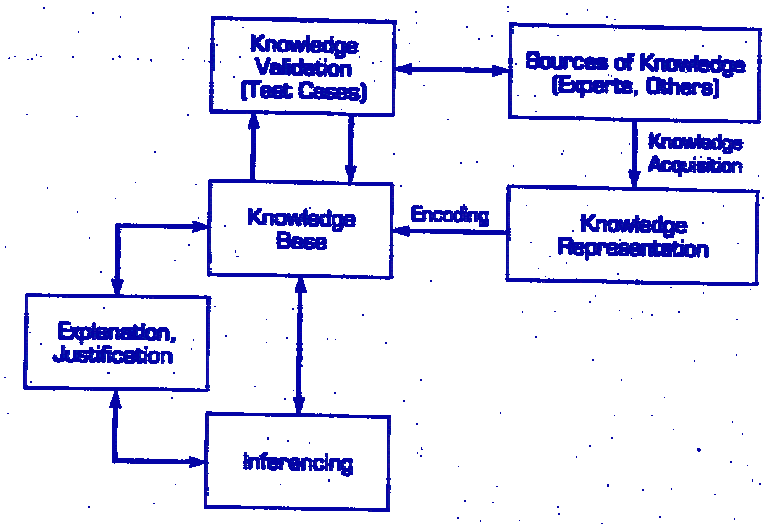
Real-time Issues

- Real-time performance
- Reasoning over time-stamped data (temporal reasoning)
- Time-triggered rules (or other constructs)
- Asynchronous event handling
- External events
- Focus of attention mechanism
- Integration with real-time clock
- Bounded response time
- Continuous operation

Expert System Shells

- CLIPS
  - a productive development and delivery expert system tool which provides a complete environment written in C.
  - Knowledge Representation: support for three different programming paradigms:
    - rule-based: allows knowledge to be represented as heuristics which specify a set of actions to be performed for a given situation
    - object-oriented: allows complex systems to be modeled as modular components (which can be easily reused to model other systems or to create new components).
    - procedural: similar to capabilities found in languages such as C, Java, Ada, and LISP.
- JESS
  - rule engine and scripting environment written in JAVA

Process of Knowledge Engineering



Knowledge Elicitation

- **Knowledge Elicitation** is the interaction between an expert and a knowledge engineer/program to:
  - elicit knowledge from expert in a **systematic way**
  - store knowledge so obtained in some intermediate representation
  - **compile** knowledge from intermediate representation into an **executable** form (e.g. production rules)

Example

- Task: Build expert system for dealing with oil and chemical spills in a chemical plant.
  - Become familiar with the problem and the domain (books, people)
  - Find an expert who is willing to collaborate.
  - Characterize tentatively the types of general reasoning tasks that the system is likely to perform.
  - "When an accidental inland spill of an oil or chemical occurs, an emergency situation may exist, depending on the properties and quantity of the substance released, the location of the substance, and whether or not the substance enters a body of water. The observer of a spill should 1) characterize the spill and the probable hazards, 2) contain the spilled material, 3)locate the source of the spill and stop any further release, and 4) notify the Department of Environmental Management." [Government report]
  - Make interview with the expert

An Example Interview (Early Stage)

Knowledge Engineer: Suppose you were told that a spill had been detected in White Oak Creek one mile before it enters White Oak Lake. What would you do to contain the spill.

Expert: That depends on a number of factors. I would need to find the source in order to prevent the possibility of further contamination, probably by checking drains and manholes for signs of the spill material. And it helps to know what the spilled material is.

Knowledge Engineer: How can you tell what it is?

Expert: Sometimes you can tell what the substance is by its smell. Sometimes you can tell by its color, but that's not always reliable since dyes are used a lot nowadays. Oil, however, oats on the surface and forms a silvery film, while acids dissolve completely in the water. Once you discover the type of material spilled, you can eliminate any buildings that either don't store the material at all or don't store enough of it to account for the spill.

What to Look for?

- The knowledge engineer listens in order to:
  - characterize the expertise in broad kinds of knowledge
  - note technical terms (like color, odor), classify them
  - listen to the basic strategies the expert uses: What facts does the expert try to establish first? What kinds of questions does the expert ask first? Does the expert make initial guesses about anything based on tentative information? How does the expert then determine which question to use to refine the guess?
  - listen to justifications of the associations, terms, and strategic methods.
  - ask additional questions for clarification.

Typical Result of Conceptualization

- Task: Identification of spill material
  - Attribute of spill:
    - Type of spill: Oil, acid
    - Location of spill <A set of drains & manholes>
    - Volume of spill <A number of liters>
  - Attributes of material:
    - Color: Silvery, clear, etc.
    - Odor: Pungent/choking, etc.
    - Does it dissolve?
    - Possible locations: <A set of buildings>
    - Amount stored: <A number of liters>
- Some of the terms are deliberately ignored in order to simplify the task (e.g. reliability and dyes)

Encoding the Example

- Further formalize the case knowledge in form of primitive concepts like
  - Assert each of BUILDING-3023 & BUILDING-3024 is a building
  - Assert s6-1 is a source in BUILDING-3023
  - Assert s6-1 holds 2000 liters of gasoline
  - Assert every oil is a possible material of the spill and every acid is a possible material of the spill
  - Assert the spill smells of [some odor, e.g., a pungent/choking, no] odor
  - Assert the spill [does, does not] dissolve in water.

Encoding the Example (Cont'd)

Formalize the expert knowledge, e.g. to determine the spill material in rules like

- IF: the spill does not dissolve in water and the spill does form a silvery film
  THEN: let the spill be oil
- IF: the spill does dissolve in water and the spill does form no lm
  THEN: let the spill be acid
- IF: the spill=oil and the odor of the spill is known
  THEN: choose situation:
  - IF: the spill does smell of gasoline
    THEN: let the material of the spill be gasoline with certainty 0.9
  - IF: the spill does smell of diesel oil
    THEN: let the material of the spill be diesel oil with certainty 0.8

Testing

Knowledge Engineer: Here are some rules I think capture your explanation about determining the type of material spilled and eliminating possible spill sources. What do you think?

Expert: Uh-huh (long pause). Yes, that begins to capture it. Of course if the material is silver nitrate it will dissolve only partially in the water.

Knowledge Engineer: I see. Well, let's add that information to the knowledge base and see what it looks like.

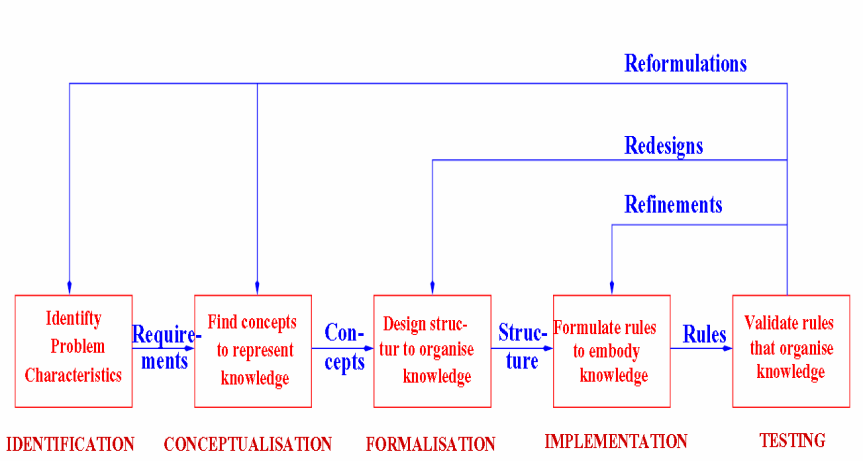
Add: Assert the solubility of the spill is [some level-high, moderate, low]

Delete: Assert the spill [does, does not] dissolve in water.

Modify: (1) If the solubility of the spill is low and the spill does form a silvery film, let the spill be oil.

Add: (1.5) If the solubility of the spill is moderate, let the material of the spill be silver-nitrate with certainty 0.6

Stages of Knowledge Acquisition



Levels in the Analysis of Knowledge

- Knowledge identification: State of in-depth interview, in which a knowledge engineer encourages experts to talks about how they do what they do. In Interview decide what to talk about. Prerequisite: understand the domain well enough to know which objects and facts need to be talked about.
- Knowledge conceptualization: find primitive concepts and conceptual relations.
- Epistemological analysis: uncover structural properties of the conceptual knowledge, such as taxonomic relations.
- Logical analysis: how to perform reasoning in the domain. This kind of knowledge is particularly hard to acquire.
- Implementational analysis: implementing and testing.

Interviewing Strategies

- Descriptive Elicitation:

Type	Example
Grand Tour Questions	"Could you tell me about a typical telephone interview"
	"Tell about typical open-ended questions"
	"Could you turn through the survey form and point out the different parts?"
Case-Focused Questions	"Can you give me an example of `probing for a more specific response'?"
Language Questions	"What would you call a respondent who kept saying he was going to hang up?"

Tacit Knowledge

- Problem: Experts use implicit knowledge (e.g. procedural knowledge) that is difficult to capture from human experts
- Techniques for acquiring this knowledge
  - Protocol analysis: Tape-record thinking aloud and later analyze this: break down the subjects' protocol into smallest, atomic units of thought, these become operators
  - Participant observation, knowledge engineer acquires tacit knowledge through practical domain experience
  - Machine induction, useful since experts are usually able to supply examples of their decision making, even if they cannot articulate the underlying knowledge.

Protocol Analysis

- Method of studying subjects' mental processes in the performance of tasks. [Newell & Simon 1972]
- Framework:
  - Expert viewed as processor producing certain problem-solving behavior.
  - Expert has a set of actions and abilities necessary to realize this expertise.
  - Expertise is not directly observable, but expert's actions and abilities are.
  - Develop a representation of the expertise from invocation of actions and abilities
  - A statement of the expertise required to perform a task serves as a requirement for the computer program designed to perform that task

Maxims for Finding Rules

- Don't just talk with the expert, watch him/her doing examples
- Use the terms and methods that the experts uses.
- Look at intermediate-level abstractions.
- If a rule looks big, it is
- If several rules are very similar, look for an underlying domain concept.
- If tempted to escape from knowledge representation formalism into pure code, resist the temptation [for at least a while].

Evaluation

- Knowledge elicitation is the bottleneck of expert system technology.
- Acquisition by learning (learn factual knowledge as well as strategic knowledge)