Lecture 2

Rule-based expert systems

- Introduction, or what is knowledge?
- Rules as a knowledge representation technique
- The main players in the development team
- Structure of a rule-based expert system
- Characteristics of an expert system
- Forward chaining and backward chaining
- Conflict resolution
- Summary

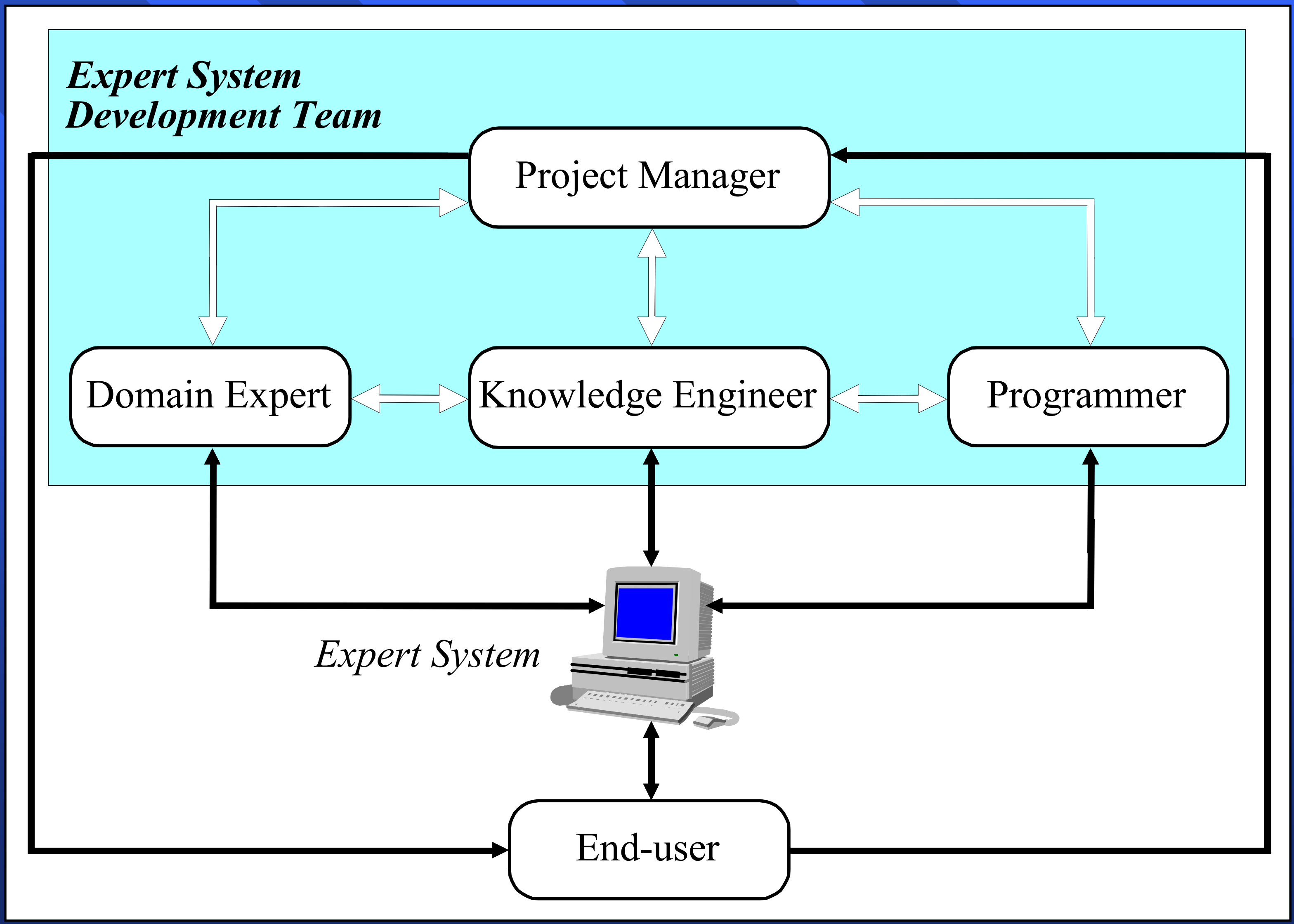
Rules as a knowledge representation technique

- The term ***rule*** in AI, which is the most commonly used type of knowledge representation, can be defined as an IF-THEN structure that relates given information or facts in the IF part to some action in the THEN part. A rule provides some description of how to solve a problem. Rules are relatively easy to create and understand.
- Any rule consists of two parts: the IF part, called the ***antecedent*** (***premise*** or ***condition***) and the THEN part called the ***consequent*** (***conclusion*** or ***action***).

- The antecedent of a rule incorporates two parts: an ***object*** (*linguistic object*) and its ***value***. The object and its value are linked by an ***operator***.
- The operator identifies the object and assigns the value. Operators such as *is*, *are*, *is not*, *are not* are used to assign a **symbolic value** to a linguistic object.
- Expert systems can also use mathematical operators to define an object as numerical and assign it to the **numerical value**.

IF	'age of the customer' < 18
AND	'cash withdrawal' > 1000
THEN	'signature of the parent' is required

The main players in the development team



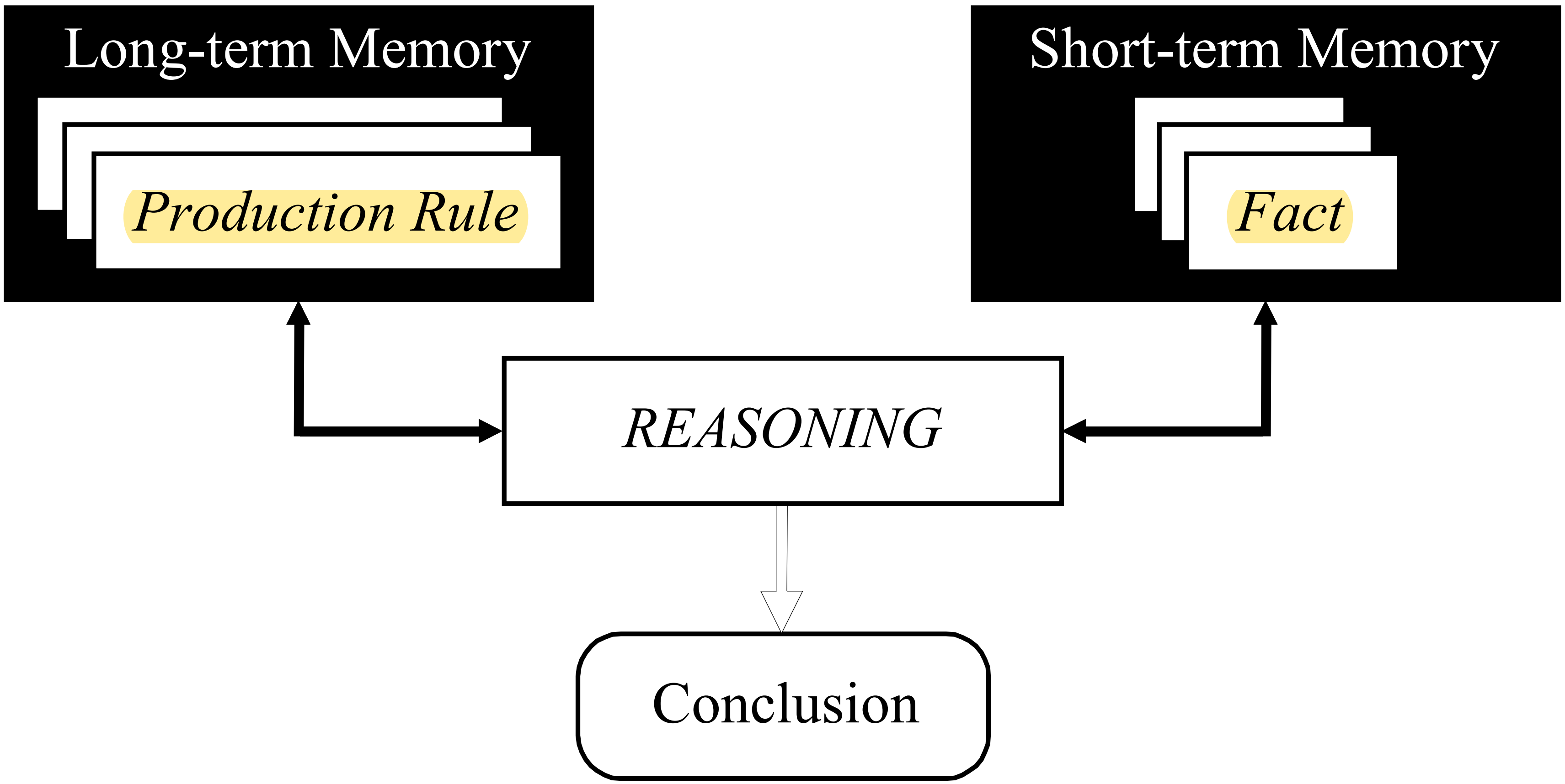
- The ***domain expert*** is a knowledgeable and skilled person capable of solving problems in a specific area or ***domain***. This person has the greatest expertise in a given domain. This expertise is to be captured in the expert system. Therefore, the expert must be able to communicate his or her knowledge, be willing to participate in the expert system development and commit a substantial amount of time to the project. The domain expert is the most important player in the expert system development team.

- The ***knowledge engineer*** is someone who is capable of designing, building and testing an expert system. He or she interviews the domain expert to find out how a particular problem is solved. The knowledge engineer establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system. The knowledge engineer then chooses some development software or an expert system shell, or looks at programming languages for encoding the knowledge. And finally, the knowledge engineer is responsible for testing, revising and integrating the expert system into the workplace.

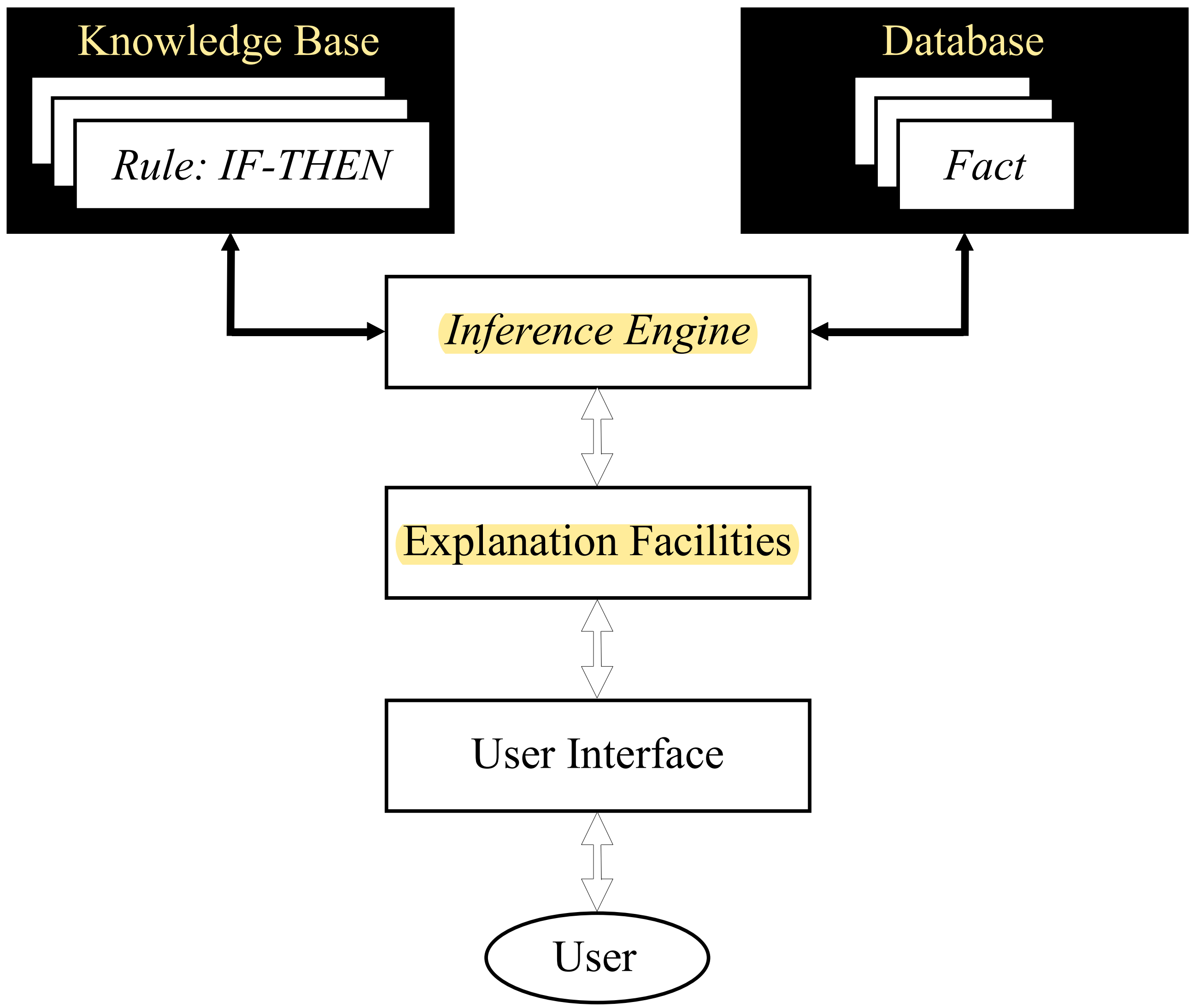
- The ***programmer*** is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand. The programmer needs to have skills in symbolic programming in such AI languages as LISP, Prolog and OPS5 and also some experience in the application of different types of expert system shells. In addition, the programmer should know conventional programming languages like C, Pascal, FORTRAN and Basic.

- The ***project manager*** is the leader of the expert system development team, responsible for keeping the project on track. He or she makes sure that all deliverables and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user.
- The ***end-user***, often called just the *user*, is a person who uses the expert system when it is developed. The user must not only be confident in the expert system performance but also feel comfortable using it. Therefore, the design of the user interface of the expert system is also vital for the project's success; the end-user's contribution here can be crucial.

Production system model



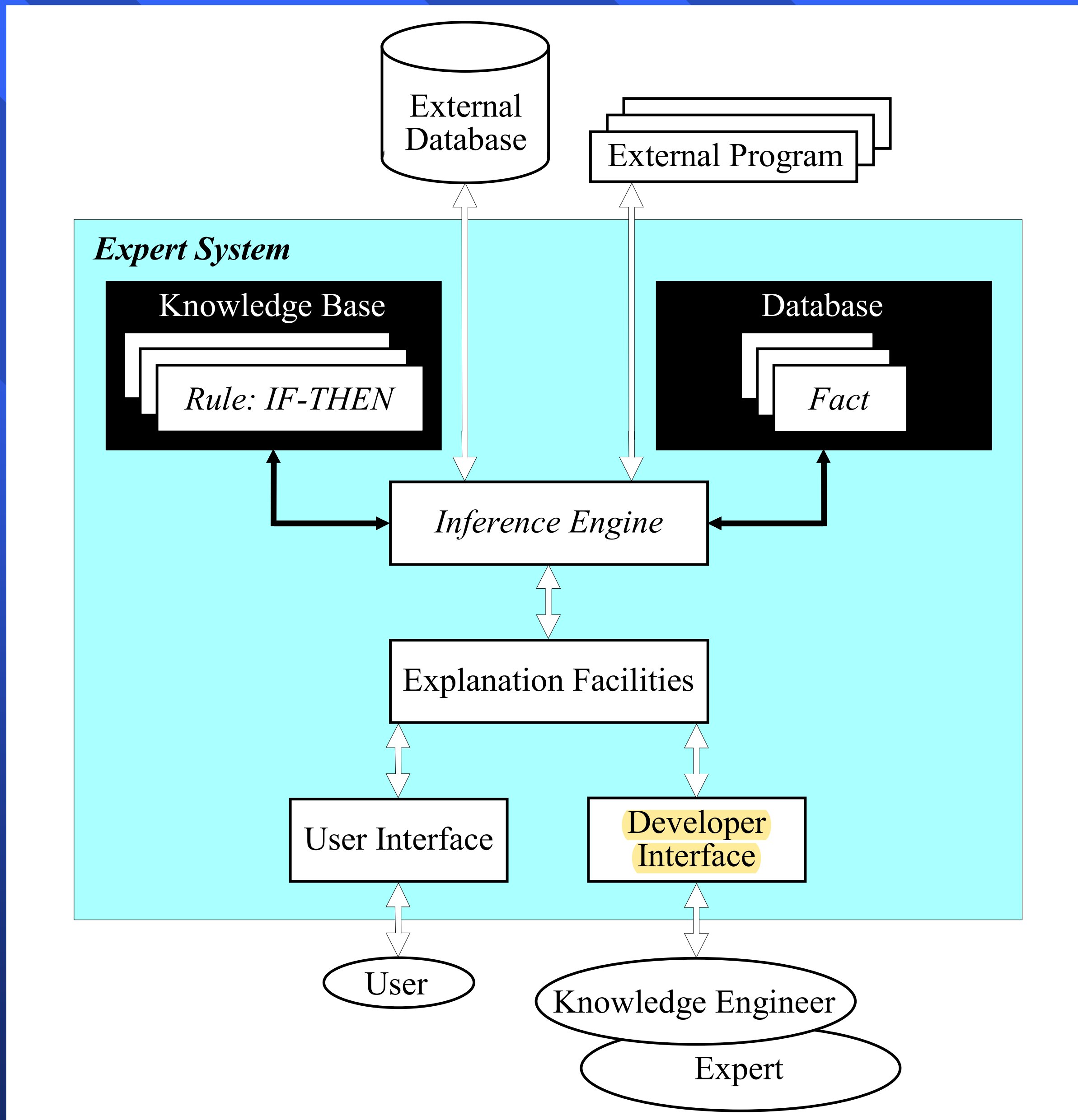
Basic structure of a rule-based expert system



- The **knowledge base** contains the domain knowledge useful for problem solving. In a rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the IF (condition) THEN (action) structure. When the condition part of a rule is satisfied, the rule is said to ***fire*** and the action part is executed.
- The **database** includes a set of facts used to match against the IF (condition) parts of rules stored in the knowledge base.

- The **inference engine** carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database.
- The **explanation facilities** enable the user to ask the expert system ***how*** a particular conclusion is reached and ***why*** a specific fact is needed. An expert system must be able to explain its reasoning and justify its advice, analysis or conclusion.
- The **user interface** is the means of communication between a user seeking a solution to the problem and an expert system.

Complete structure of a rule-based expert system



Comparison of expert systems with conventional systems and human experts

Human Experts	*Expert Systems*	*Conventional Programs*
Use knowledge in the form of rules of thumb or heuristics to solve problems in a narrow domain.	Process knowledge expressed in the form of rules and use symbolic reasoning to solve problems in a *narrow domain*.	Process data and use algorithms, a series of well-defined operations, to solve general numerical problems.
In a human brain, knowledge exists in a compiled form.	Provide a *clear separation of knowledge from its processing*.	Do not separate knowledge from the control structure to process this knowledge.
Capable of explaining a line of reasoning and providing the details.	*Trace the rules fired* during a problem-solving session and *explain how* a particular conclusion was reached and *why* specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.

Comparison of expert systems with conventional systems and human experts (*Continued*)

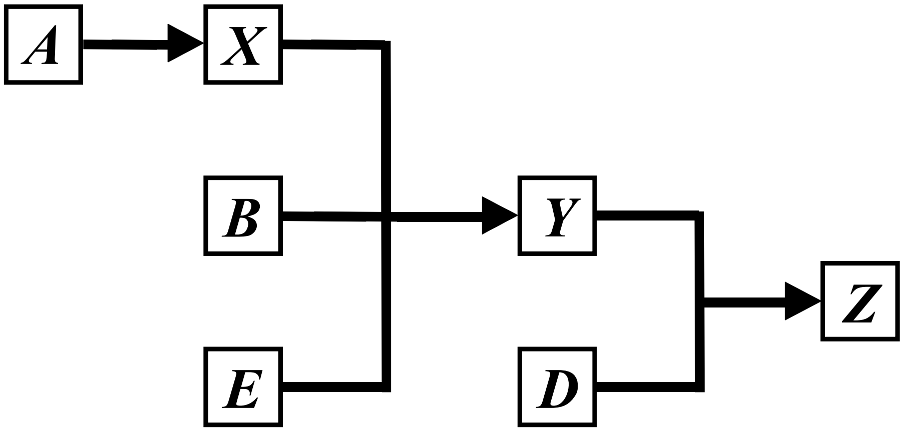
Human Experts	*Expert Systems*	*Conventional Programs*
Use inexact reasoning and can deal with incomplete, uncertain and fuzzy information.	Permit *inexact reasoning* and can deal with incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Can make mistakes when information is incomplete or fuzzy.	*Can make mistakes* when data is incomplete or fuzzy.	Provide no solution at all, or a wrong one, when data is incomplete or fuzzy.
Enhance the quality of problem solving via years of learning and practical training. This process is slow, inefficient and expensive.	Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, *changes are easy* to accomplish.	Enhance the quality of problem solving by changing the program code, which affects both the knowledge and its processing, making changes difficult.

An example of an inference chain

Rule 1: IF *Y* is true
AND *D* is true
THEN *Z* is true

Rule 2: IF *X* is true
AND *B* is true
AND *E* is true
THEN *Y* is true

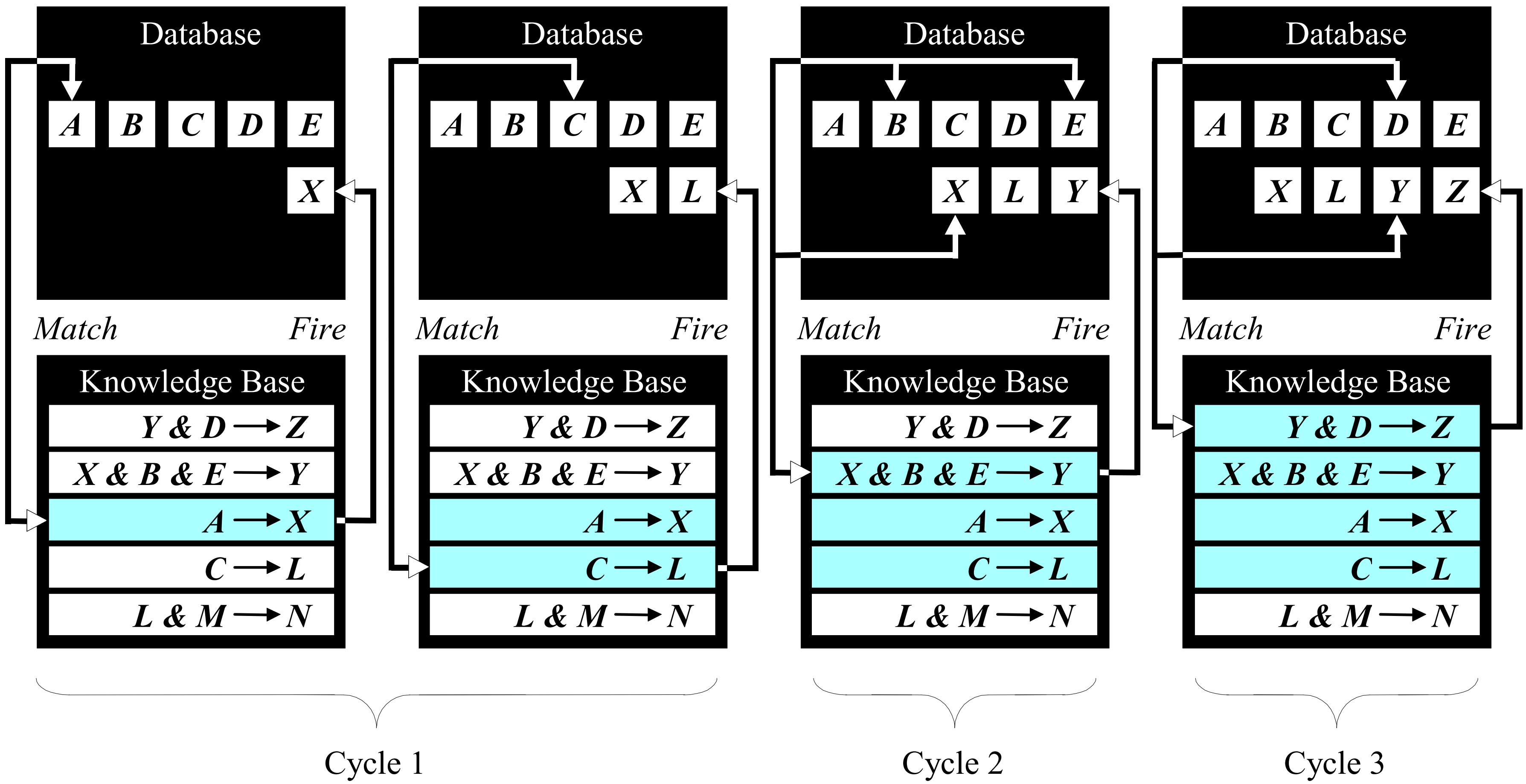
Rule 3: IF *A* is true
THEN *X* is true



Forward chaining

- Forward chaining is the **data-driven reasoning**. The reasoning starts from the known data and proceeds forward with that data. Each time only the topmost rule is executed. When fired, the rule adds a new fact in the database. Any rule can be executed only once. The match-fire cycle stops when no further rules can be fired.

Forward chaining



- Forward chaining is a technique for gathering information and then inferring from it whatever can be inferred.
- However, in forward chaining, many rules may be executed that have nothing to do with the established goal.
- Therefore, if our goal is to infer only one particular fact, the forward chaining inference technique would not be efficient.

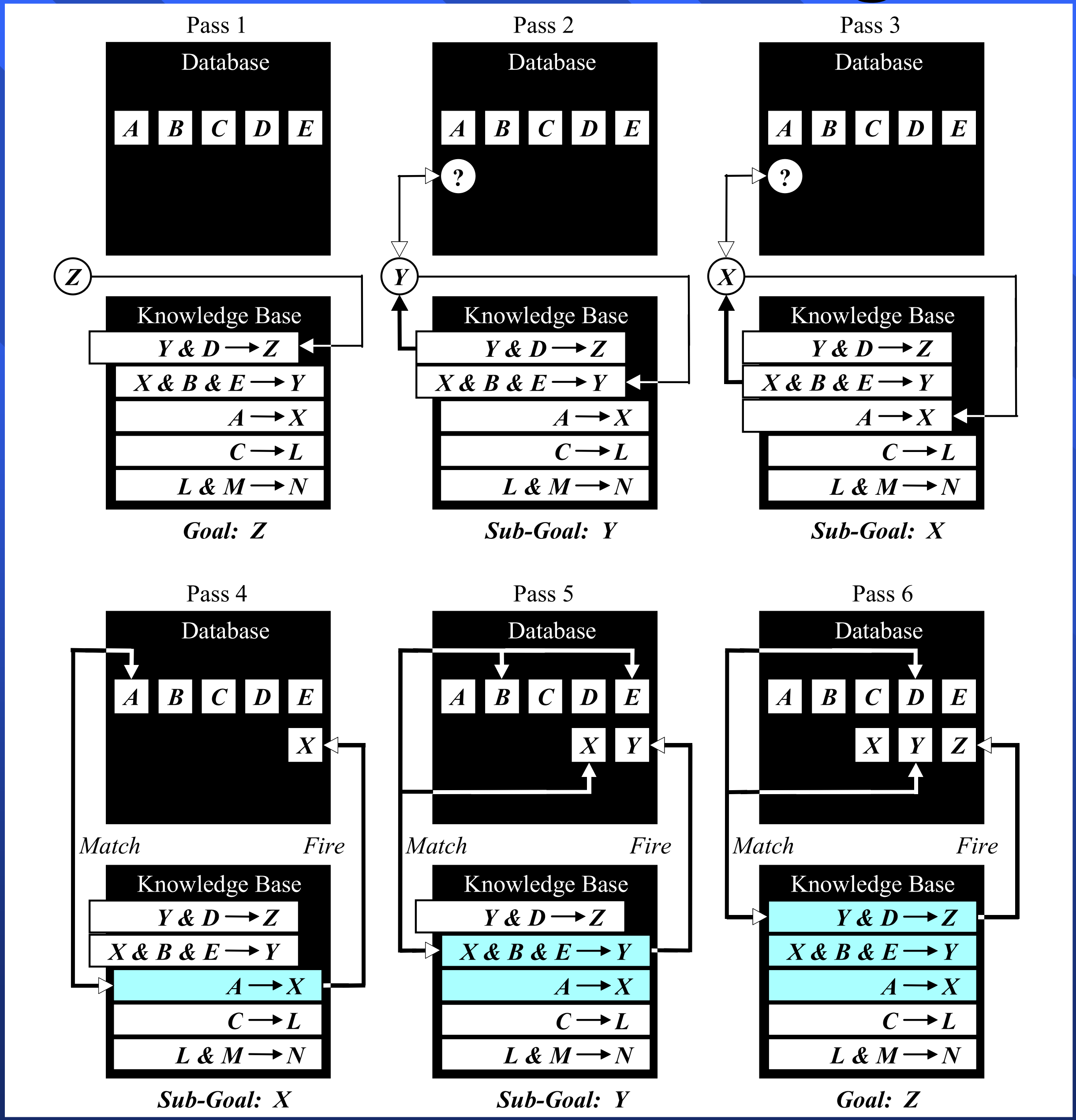
Backward chaining

- Backward chaining is the **goal-driven reasoning**. In backward chaining, an expert system has the goal (a *hypothetical solution*) and the inference engine attempts to find the evidence to prove it. First, the knowledge base is searched to find rules that might have the desired solution. Such rules must have the goal in their THEN (action) parts. If such a rule is found and its IF (condition) part matches data in the database, then the rule is fired and the goal is proved. However, this is rarely the case.

Backward chaining

- Thus the inference engine puts aside the rule it is working with (the rule is said to ***stack***) and sets up a new goal, a subgoal, to prove the IF part of this rule. Then the knowledge base is searched again for rules that can prove the subgoal. The inference engine repeats the process of stacking the rules until no rules are found in the knowledge base to prove the current subgoal.

Backward chaining



How do we choose between forward and backward chaining?

- If an expert first needs to gather some information and then tries to infer from it whatever can be inferred, choose the forward chaining inference engine.
- However, if your expert begins with a hypothetical solution and then attempts to find facts to prove it, choose the backward chaining inference engine.

Conflict resolution

Earlier we considered two simple rules for crossing a road. Let us now add third rule:

- *Rule* 1:
  IF the 'traffic light' is green
  THEN the action is go
- *Rule* 2:
  IF the 'traffic light' is red
  THEN the action is stop
- *Rule* 3:
  IF the 'traffic light' is red
  THEN the action is go

Methods used for conflict resolution

- Fire the rule with the ***highest priority***. In simple applications, the priority can be established by placing the rules in an appropriate order in the knowledge base. Usually this strategy works well for expert systems with around 100 rules.
- Fire the ***most specific rule***. This method is also known as the ***longest matching strategy***. It is based on the assumption that a specific rule processes more information than a general one.

- Fire the rule that uses the ***data most recently entered*** in the database. This method relies on time tags attached to each fact in the database. In the conflict set, the expert system first fires the rule whose antecedent uses the data most recently added to the database.

Advantages of rule-based expert systems

- **Natural knowledge representation**. An expert usually explains the problem-solving procedure with such expressions as this: "In such-and-such situation, I do so-and-so". These expressions can be represented quite naturally as IF-THEN production rules.
- **Uniform structure**. Production rules have the uniform IF-THEN structure. Each rule is an independent piece of knowledge. The very syntax of production rules enables them to be self-documented.

Advantages of rule-based expert systems

- **Separation of knowledge from its processing**. The structure of a rule-based expert system provides an effective separation of the knowledge base from the inference engine. This makes it possible to develop different applications using the same expert system shell.
- ***Dealing with incomplete and uncertain knowledge***. Most rule-based expert systems are capable of representing and reasoning with incomplete and uncertain knowledge.

Disadvantages of rule-based expert systems

- **Opaque relations between rules**. Although the individual production rules are relatively simple and self-documented, their logical interactions within the large set of rules may be opaque. Rule-based systems make it difficult to observe how individual rules serve the overall strategy.
- **Ineffective search strategy**. The inference engine applies an exhaustive search through all the production rules during each cycle. Expert systems with a large set of rules (over 100 rules) can be slow, and thus large rule-based systems can be unsuitable for real-time applications.

Disadvantages of rule-based expert systems

- ***Inability to learn***. In general, rule-based expert systems do not have an ability to learn from the experience. Unlike a human expert, who knows when to "break the rules", an expert system cannot automatically modify its knowledge base, or adjust existing rules or add new ones. The knowledge engineer is still responsible for revising and maintaining the system.