Unit 4: Rule Based Expert System

Prepared by Prof.Kore S.S

(Assistant Professor(CSE) DYPCET,Kolhapur)

Expert system development team

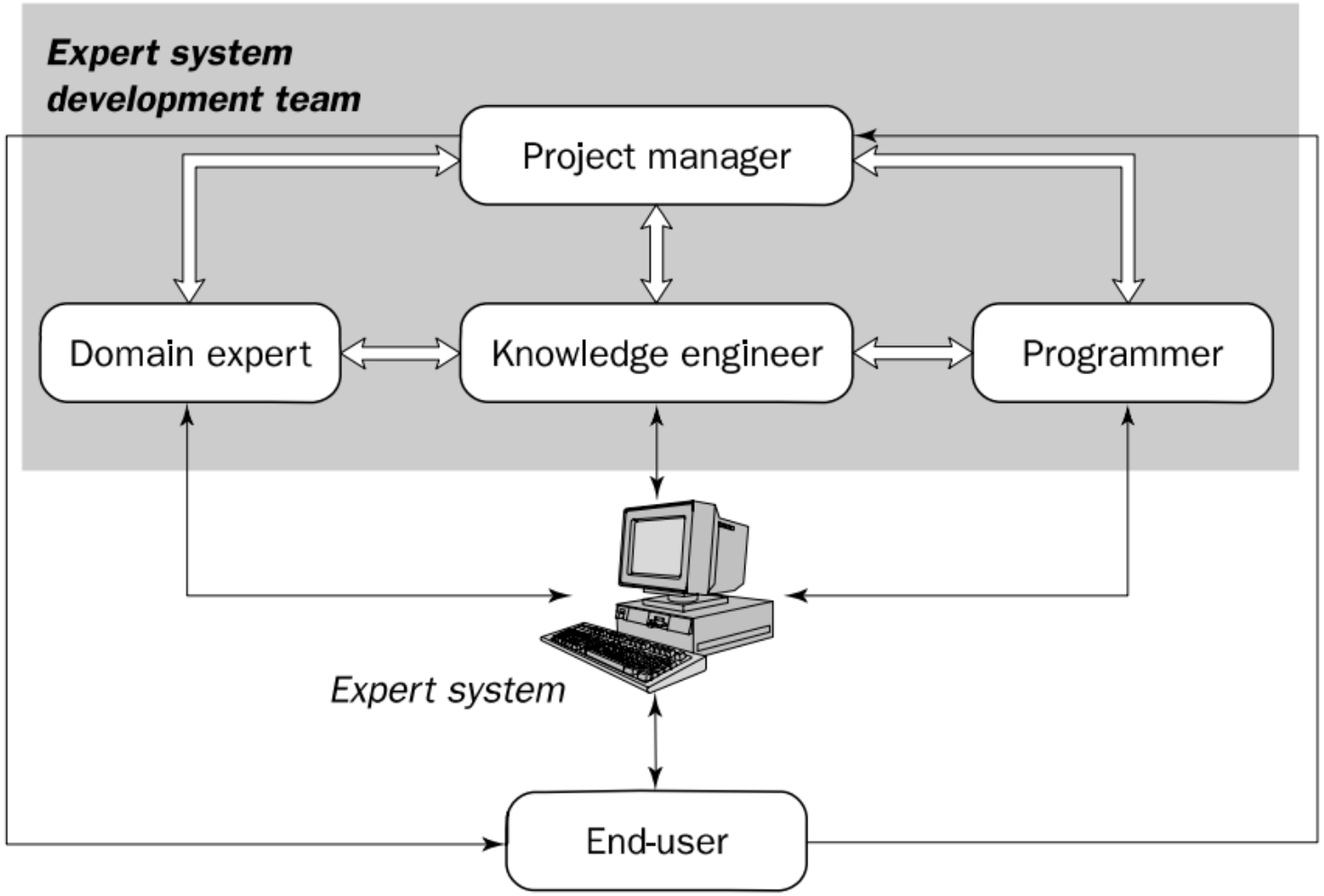


Figure 2.1 The main players of the expert system development team

There are five members of the expert system development team: the domain expert, the knowledge engineer, the programmer, the project manager and the end-user

1.Domain Expert

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.

2.Knowledge Engineer

- The knowledge engineer is someone who is capable of designing, building and testing an expert system. This person is responsible for selecting an appropriate task for the expert system.
- The knowledge engineer is responsible for testing, revising and integrating the expert system into the workplace. Thus, the knowledge engineer is committed to the project from the initial design stage to the final delivery of the expert system, and even after the project is completed, he or she may also be involved in maintaining the system.

3.Programmer

- The programmer is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand

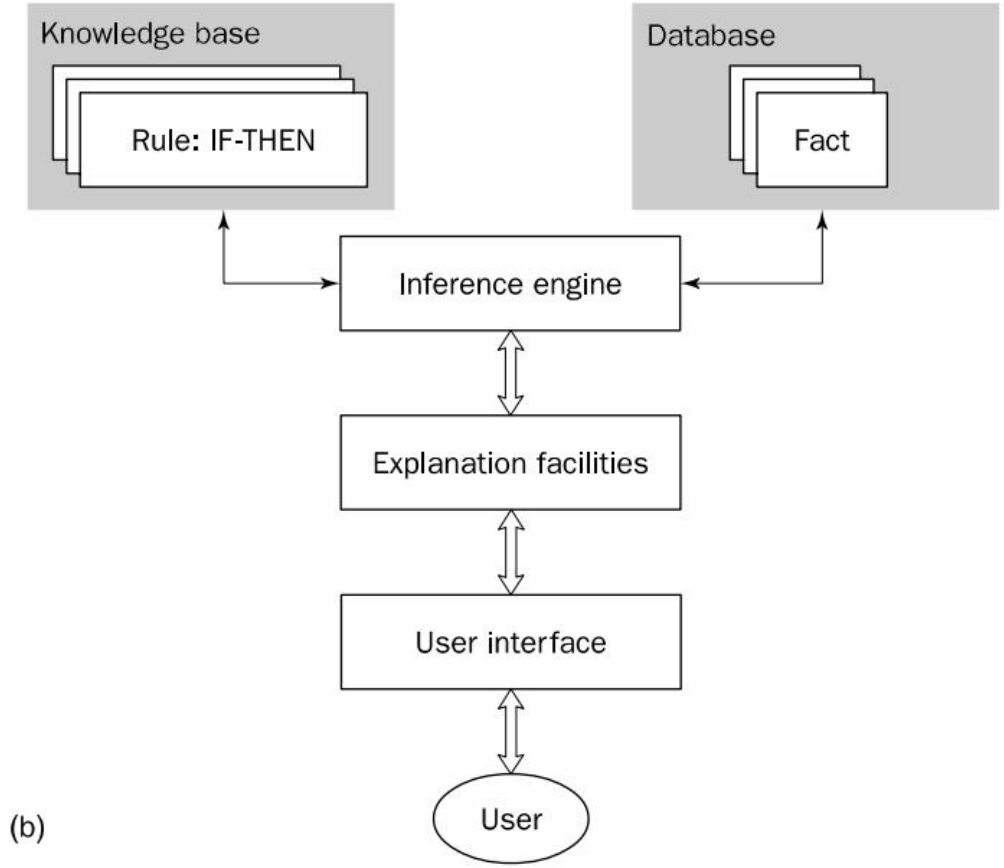
4.Project manager

- 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.

5.End user

- The end-user, often called just the user, is a person who uses the expert system when it is developed.

Structure of a rule-based expert system



(b)

1.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.

2.The database includes a set of facts used to match against the IF (condition) parts of rules stored in the knowledge base.

3.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

4.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.

5.The user interface is the means of communication between a user seeking a solution to the problem and an expert system. The communication should be as meaningful and friendly as possible

Characteristics of an expert system

1.Narrow, specialised domain

An expert system is built to perform at a human expert level in a narrow, specialized domain.

No matter how fast the system can solve a problem, the user will not be satisfied if the result is wrong. On the other hand, the speed of reaching a solution is very important. Even the most accurate decision or diagnosis may not be useful if it is too late to apply, for instance, in an emergency, when a patient dies or a nuclear power plant explodes.

2.Explanation capability.

This enables the expert system to review its own reasoning and explain its decisions. An explanation in expert systems in effect traces the rules fired during a problem-solving session.

3.Symbolic reasoning

Expert systems employ symbolic reasoning when solving a problem. Symbols are used to represent different types of knowledge such as facts, concepts and rules.

4. Make mistakes

Expert system can make mistakes when data is incomplete or fuzzy.

5.Knowledge is separated from its processing

Important feature that distinguishes expert systems from conventional programs is that knowledge is separated from its processing (the knowledge base and the inference engine are split up). A conventional program is a mixture of knowledge and the control structure to process this knowledge.

Inference chains.

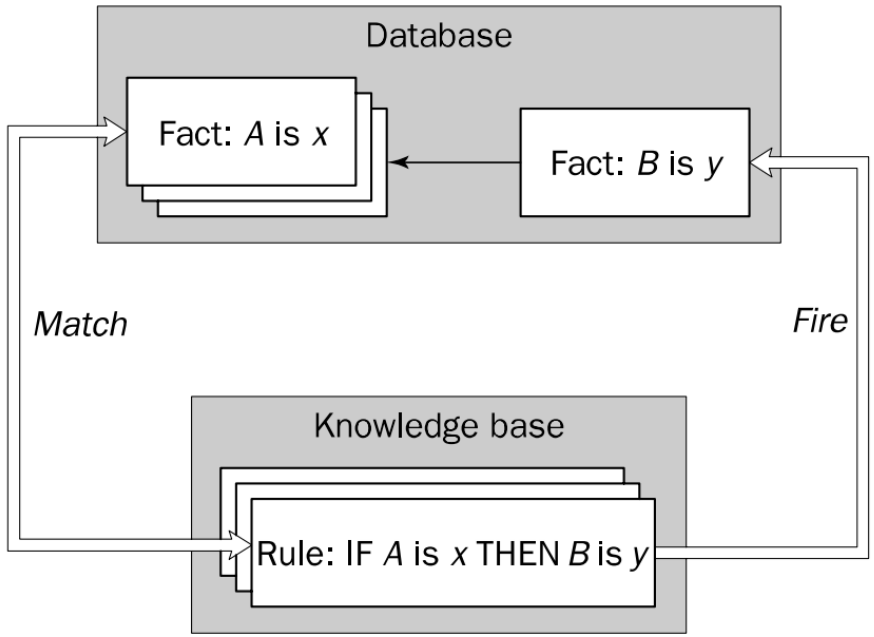


Figure 2.4 The inference engine cycles via a match-fire procedure

When the IF (condition) part of the rule matches a fact, the rule is fired and its THEN (action) part is executed.

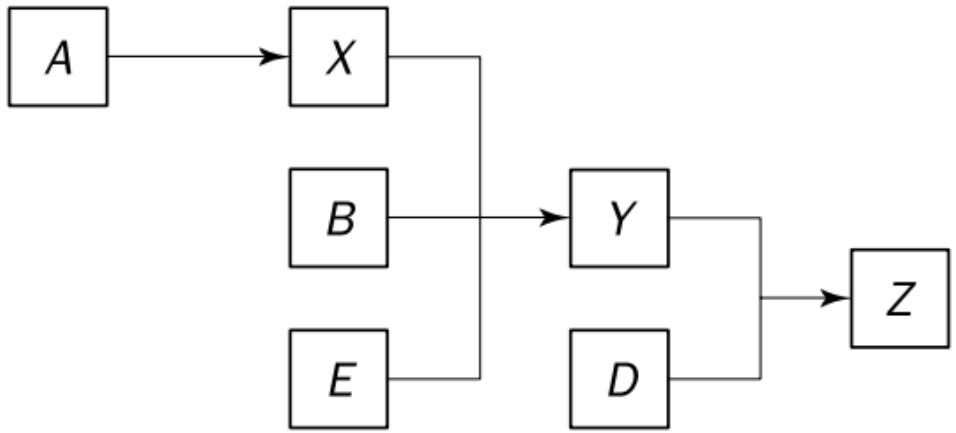


Figure 2.5 An example of an inference chain

Forward chaining

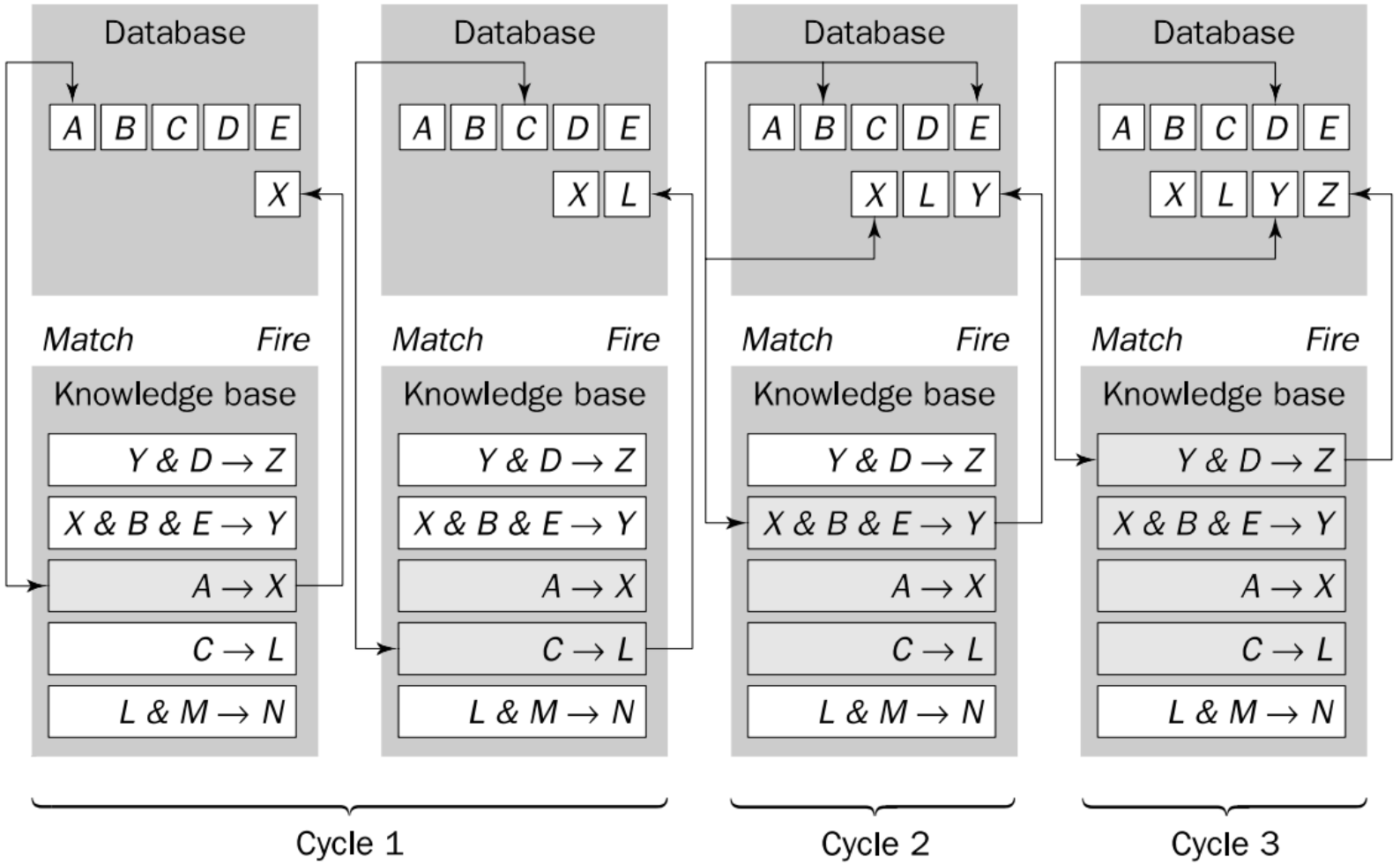


Fig 2.6

- 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.
- In the first cycle, only two rules, Rule 3: A→X and Rule 4: C→L, match facts in the database. Rule 3: A→X is fired first as the topmost one. The IF part of this rule matches fact A in the database, its THEN part is executed and new fact X is added to the database. Then Rule 4: C→ L is fired and fact L is also placed in the database.
- In the second cycle, Rule 2: X & B & E→Y is fired because facts B, E and X are already in the database, and as a consequence fact Y is inferred and put in the database. This in turn causes Rule 1: Y & D→Z to execute, placing fact Z in the database (cycle 3). Now the match-fire cycles stop because the IF part of Rule 5: L & M→N does not match all facts in the database and thus Rule 5 cannot be fired.

Backward chaining

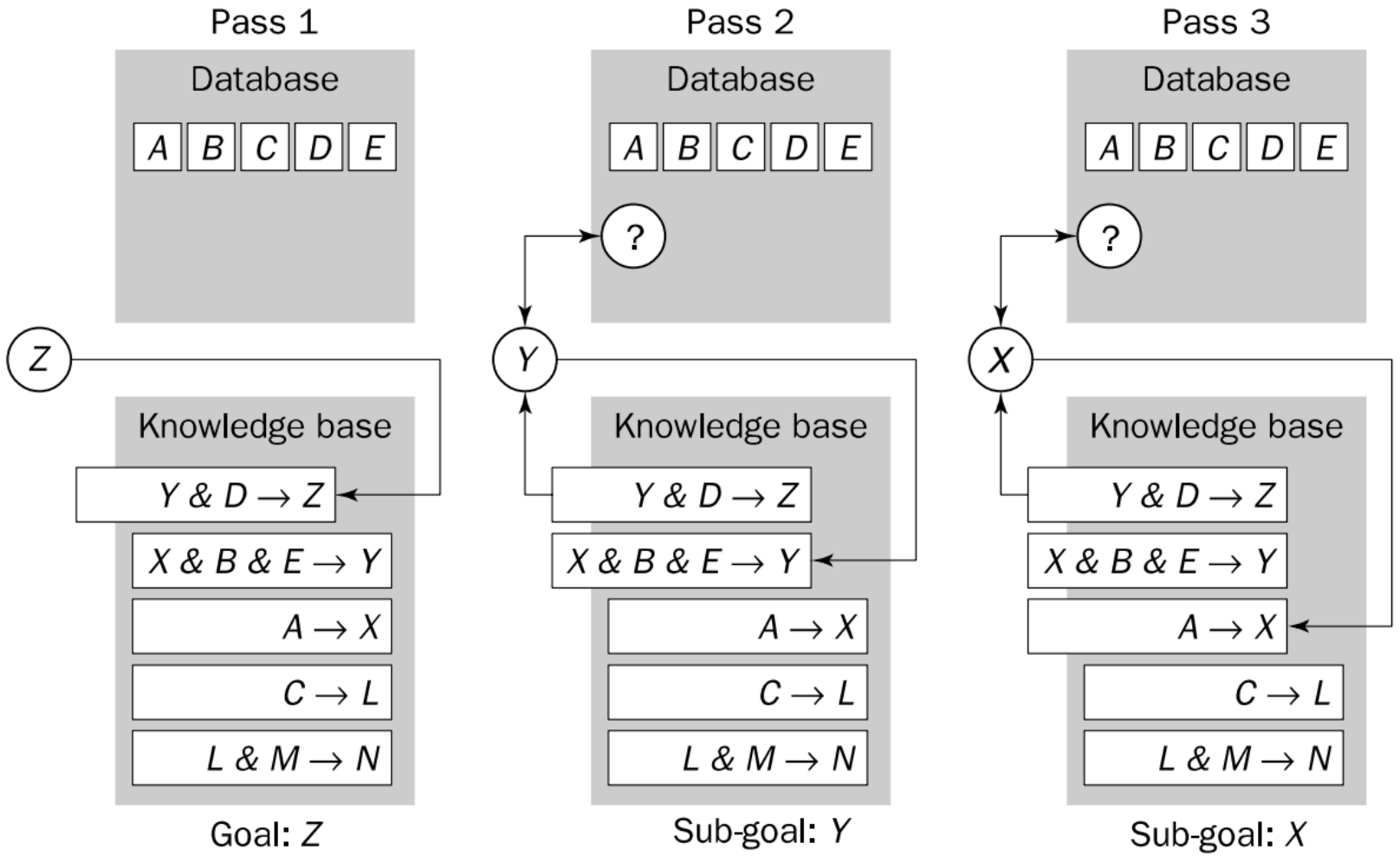


Fig 2.7

- In Pass 1, the inference engine attempts to infer fact Z. It searches the knowledge base to find the rule that has the goal, in our case fact Z, in its THEN part. The inference engine finds and stacks Rule 1: Y & D -> Z. The IF part of Rule 1 includes facts Y and D, and thus these facts must be established.

- In Pass 2, the inference engine sets up the sub-goal, fact Y, and tries to determine it. First it checks the database, but fact Y is not there. Then the knowledge base is searched again for the rule with fact Y in its THEN part. The inference engine locates and stacks Rule 2: X & B & E -> Y. The IF part of Rule 2 consists of facts X, B and E, and these facts also have to be established.

- In Pass 3, the inference engine sets up a new sub-goal, fact X. It checks the database for fact X, and when that fails, searches for the rule that infers X. The inference engine finds and stacks Rule 3: A ->X. Now it must determine fact A

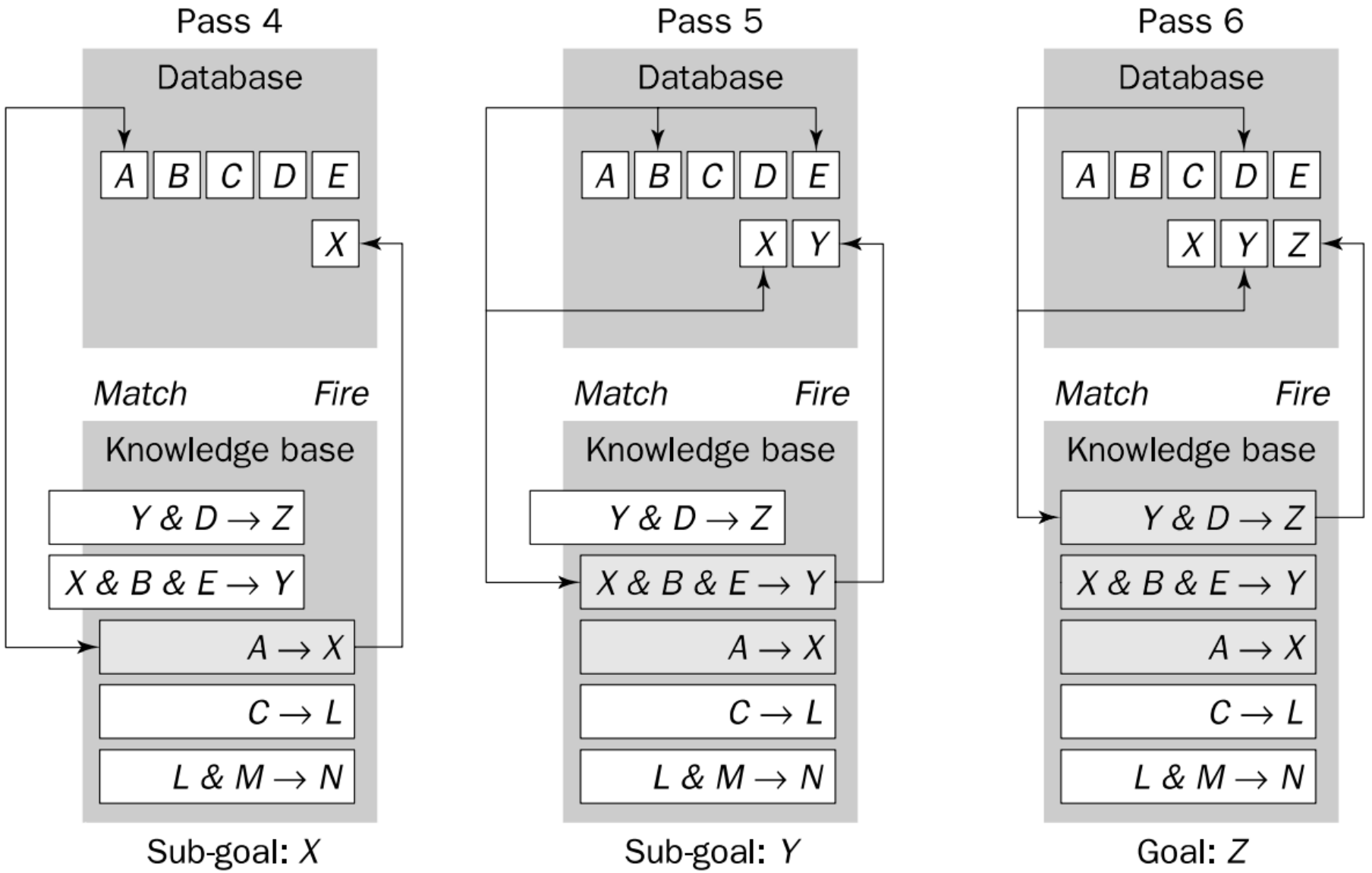

Pass 4
Database
A B C D E
X
Match
Fire
Knowledge base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N
Sub-goal: X
Pass 5
Database
A B C D E
X Y
Match
Fire
Knowledge base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N
Sub-goal: Y
Pass 6
Database
A B C D E
X Y Z
Match
Fire
Knowledge base
Y & D → Z
X & B & E → Y
A → X
C → L
L & M → N
Goal: Z

Fig 2.7

- In Pass 4, the inference engine finds fact A in the database, Rule 3: A ->X is fired and new fact X is inferred.

- In Pass 5, the inference engine returns to the sub-goal fact Y and once again tries to execute Rule 2: X & B & E -> Y. Facts X, B and E are in the database and thus Rule 2 is fired and a new fact, fact Y, is added to the database.

- In Pass 6, the system returns to Rule 1: Y & D -> Z trying to establish the original goal, fact Z. The IF part of Rule 1 matches all facts in the database, Rule 1 is executed and thus the original goal is finally established.

- Let us now compare Figure 2.6 with Figure 2.7. As you can see, four rules were fired when forward chaining was used, but just three rules when we applied backward chaining. This simple example shows that the backward chaining inference technique is more effective when we need to infer one particular fact, in our case fact Z.

MEDIA ADVISOR: a demonstration rule-based expert system

- The system provides advice on selecting a medium for delivering a training program based on the trainee's job
- if a trainee is a mechanical technician responsible for maintaining hydraulic systems, an appropriate medium might be a workshop,
- if a trainee is a clerk assessing insurance applications, a training program might include lectures on specific problems of the task, as well as tutorials where the trainee could evaluate real applications
- For complex tasks, where trainees are likely to make mistakes, a training program should also include feedback on the trainee's performance

- Knowledge base
- /* MEDIA ADVISOR: a demonstration rule-based expert system
- Rule: 1

if the environment is papers
or the environment is manuals
or the environment is documents
or the environment is textbooks
then the stimulus_situation is verbal

- Rule: 2

if the environment is pictures
or the environment is illustrations
or the environment is photographs
or the environment is diagrams
then the stimulus_situation is visual

- Rule: 3

if the environment is machines
or the environment is buildings
or the environment is tools
then the stimulus_situation is ‘physical object

Rule: 4
if the environment is numbers
or the environment is formulas
or the environment is ‘computer programs’
then the stimulus_situation is symbolic

Rule: 5
if the job is lecturing
or the job is advising
or the job is counselling
then the stimulus_response is oral

Rule: 6
if the job is building
or the job is repairing
or the job is troubleshooting
then the stimulus_response is ‘hands-on’

Rule: 7
if the job is writing
or the job is typing
or the job is drawing
then the stimulus_response is documented

Rule: 8
if the job is evaluating
or the job is reasoning
or the job is investigating
then the stimulus_response is analytical

Rule: 9
if the stimulus_situation is ‘physical object’
and the stimulus_response is ‘hands-on’
and feedback is required
then medium is workshop

Rule: 10
if the stimulus_situation is symbolic
and the stimulus_response is analytical
and feedback is required
then medium is ‘lecture – tutorial’

Rule: 11
if the stimulus_situation is visual
and the stimulus_response is documented
and feedback is not required
then medium is videocassette

Rule: 12
if the stimulus_situation is visual
and the stimulus_response is oral
and feedback is required
then medium is ‘lecture – tutorial’

Rule: 13
if the stimulus_situation is verbal
and the stimulus_response is analytical
and feedback is required
then medium is ‘lecture – tutorial’

Rule: 14
if the stimulus_situation is verbal
and the stimulus_response is oral
and feedback is required
then medium is ‘role-play exercises’

Object	Allowed values	Object	Allowed values
environment	*papers*	*job*	*lecturing*
	manuals		*advising*
	documents		*counselling*
	textbooks		*building*
	pictures		*repairing*
	illustrations		*troubleshooting*
	photographs		*writing*
	diagrams		*typing*
	machines		*drawing*
	buildings		*evaluating*
	tools		*reasoning*
	numbers		*investigating*
	formulas		
	computer programs	*stimulus_ response*	*oral*
			hands-on
			documented
			analytical
stimulus_situation	*verbal*		
	visual		
	physical object	*feedback*	*required*
	symbolic		*not required*

- An object and its value constitute a fact (for instance, the environment is machines, and the job is repairing). All facts are placed in the database
- Options

The final goal of the rule-based expert system is to produce a solution to the problem based on input data. In MEDIA ADVISOR, the solution is a medium selected from the list of four options:

- medium is workshop
- medium is ‘lecture – tutorial’
- medium is videocassette
- medium is ‘role-play exercises’

Dialogue

- Dialogue In the dialogue shown below, the expert system asks the user to input the data needed to solve the problem (the environment, the job and feedback). Based on the answers supplied by the user

- In the dialogue shown below, the expert system asks the user to input the data needed to solve the problem (the environment, the job and feedback). Based on the answers supplied by the user

What sort of environment is a trainee dealing with on the job?
⇒ **machines**

Rule: 3
if the environment is machines
or the environment is buildings
or the environment is tools
then the stimulus_situation is 'physical object'

In what way is a trainee expected to act or respond on the job?
⇒ **repairing**

Rule: 6
if the job is building
or the job is repairing
or the job is troubleshooting
then the stimulus_response is 'hands-on'

Is feedback on the trainee's progress required during training?
⇒ **required**

Rule: 9
if the stimulus_situation is 'physical object'
and the stimulus_response is 'hands-on'
and feedback is required
then medium is workshop

medium is workshop

Forward chaining is data-driven reasoning, so we need first to provide some data. Assume that

the environment is **machines**
'**environment**' instantiated by user input to '**machines**'

the job is **repairing**
'**job**' instantiated by user input to '**repairing**'

feedback is **required**
'**feedback**' instantiated by user input to '**required**'

The following process will then happen:

Rule: 3 fires	'**stimulus_situation**' instantiated by Rule: 3 to '**physical object**'
Rule: 6 fires	'**stimulus_response**' instantiated by Rule: 6 to '**hands-on**'
Rule: 9 fires	'**medium**' instantiated by Rule: 9 to '**workshop**'
No rules fire	stop

Backward chaining is goal-driven reasoning, so we need first to establish a hypothetical solution (the goal). Let us, for example, set up the following goal:

'medium' is 'workshop'

Pass 1	
Trying Rule: 9	Need to find object '**stimulus_situation**'
Rule: 9 stacked	Object '**stimulus_situation**' sought as '**physical object**'
Pass 2	
Trying Rule: 3	Need to find object '**environment**'
Rule: 3 stacked	Object '**environment**' sought as '**machines**'
ask environment	
⇒**machines**	'**environment**' instantiated by user input to '**machines**'
Trying Rule: 3	'**stimulus_situation**' instantiated by Rule: 3 to '**physical object**'
Pass 3	
Trying Rule: 9	Need to find object '**stimulus_response**'
Rule: 9 stacked	Object '**stimulus_response**' sought as '**hands-on**'

Pass 4

Trying Rule: 6	Need to find object '**job**'
Rule: 6 stacked	Object '**job**' sought as '**building**'
ask job	
⇒ **repairing**	'**job**' instantiated by user input to '**repairing**'
Trying Rule: 6	'**stimulus_response**' instantiated by Rule: 6 to '**hands-on**'

Pass 5

Trying Rule: 9	Need to find object '**feedback**'
Rule: 9 stacked	Object '**feedback**' sought as '**required**'
ask feedback	
⇒ **required**	'**feedback**' instantiated by user input to '**required**'
Trying Rule: 9	'**medium**' instantiated by Rule: 9 to '**workshop**'

medium is workshop

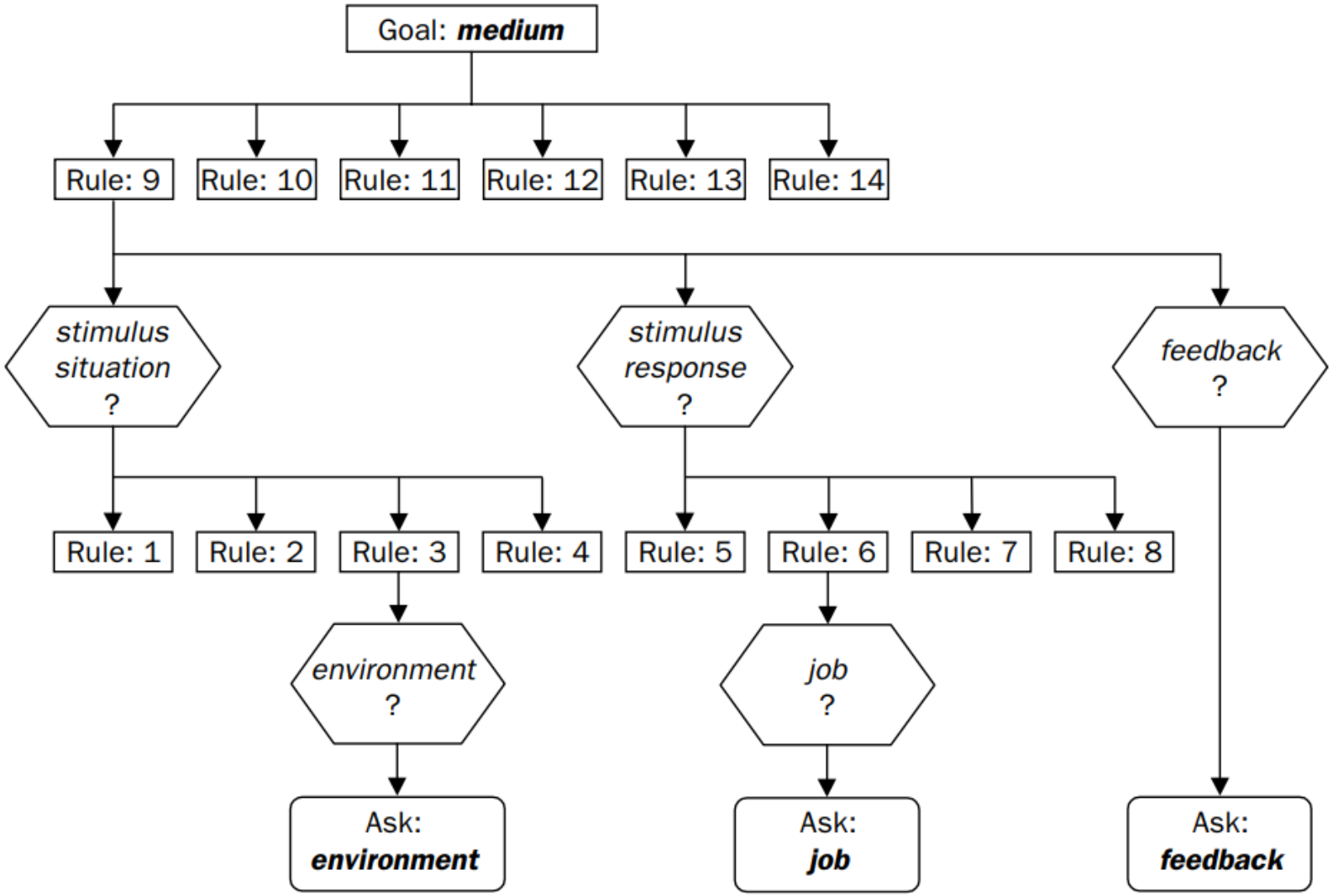


Figure 2.8 Tree diagram for the rule-based expert system MEDIA ADVISOR

Conflict resolution

- we considered two simple rules for crossing a road. Let us now add a third rule. We will get the following set of rules:

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

- The inference engine compares IF (condition) parts of the rules with data available in the database, and when conditions are satisfied the rules are set to fire. The firing of one rule may affect the activation of other rules, and therefore the inference engine must allow only one rule to fire at a time. In our road crossing example, we have two rules, Rule 2 and Rule 3, with the same IF part. Thus both of them can be set to fire when the condition part is satisfied. These rules represent a conflict set. The inference engine must determine which rule to fire from such a set. A method for choosing a rule to fire when more than one rule can be fired in a given cycle is called conflict resolution

If the traffic light is red, which rule should be executed?

- In forward chaining, both rules would be fired. Rule 2 is fired first as the topmost one, and as a result, its THEN part is executed and linguistic object action obtains value stop.
- However, Rule 3 is also fired because the condition part of this rule matches the fact ‘traffic light’ is red, which is still in the database.

  As a consequence, object action takes new value go.

How can we resolve a conflict?

1) To establish a goal and stop the rule execution when the goal is reached

- The obvious strategy for resolving conflicts is to establish a goal and stop the rule execution when the goal is reached. In our problem, for example, the goal is to establish a value for linguistic object action. When the expert system determines a value for action, it has reached the goal and stops. Thus if the traffic light is red, Rule 2 is executed, object action attains value stop and the expert system stops. In the given example, the expert system makes a right decision; however if we arranged the rules in the reverse order, the conclusion would be wrong. It means that the rule order in the knowledge base is still very important.

2)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. However, in some applications, the data should be processed in order of importance. E.g a medical consultation system

Goal 1. Prescription is? Prescription

RULE 1 Meningitis Prescription1
(Priority 100)
IF Infection is Meningitis
AND The Patient is a Child
THEN Prescription is Number_1
AND Drug Recommendation is Ampicillin
AND Drug Recommendation is Gentamicin
AND Display Meningitis Prescription1

RULE 2 Meningitis Prescription2
(Priority 90)
IF Infection is Meningitis
AND The Patient is an Adult
THEN Prescription is Number_2
AND Drug Recommendation is Penicillin
AND Display Meningitis Prescription2

3)longest matching strategy

It is based on the assumption that a specific rule processes more information than a general one. For example,

Rule 1:
IF the season is autumn
AND the sky is cloudy
AND the forecast is rain
THEN the advice is 'stay home'

Rule 2:
IF the season is autumn
THEN the advice is 'take an umbrella'

If the *season* is *autumn*, the *sky* is *cloudy* and the *forecast* is *rain*, then Rule 1 would be fired because its antecedent, the matching part, is more specific than that of Rule 2. But if it is known only that the *season* is *autumn*, then Rule 2 would be executed.

4)Data most recently entered

- 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. For example,

  Rule 1:
  IF the forecast is rain [08:16 PM 11/25/96]
  THEN the advice is 'take an umbrella'

  Rule 2:
  IF the weather is wet [10:18 AM 11/26/96]
  THEN the advice is 'stay home'

  Assume that the IF parts of both rules match facts in the database. In this case, Rule 2 would be fired since the fact *weather* is *wet* was entered after the fact *forecast* is *rain*. This technique is especially useful for real-time expert system applications when information in the database is constantly updated.

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.

- 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.

- Dealing with incomplete and uncertain knowledge: Most rule-based expert systems are capable of representing and reasoning with incomplete and uncertain knowledge. For example, the rule

```
IF      season is autumn
AND     sky is 'cloudy'
AND     wind is low
THEN    forecast is clear      { cf 0.1 };
        forecast is drizzle    { cf 1.0 };
        forecast is rain       { cf 0.9 }
```

The rule represents the uncertainty by numbers called certainty factors cf 0.1.

Disadvantages of rule-based expert systems

- Opaque relations between rules. Although the individual production rules tend to be relatively simple and self-documented, their logical interactions within the large set of rules may be opaque.

- 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

- Inability to learn:In general, rule-based expert systems do not have an ability to learn from the experience