How does goal-driven Expert System-based app work?

Why expert system-based app development?

Expert system is a classical AI technology. It simplifies app development. Avoids complex and crowded interfaces, divides interfaces into simple and manageable ones. Apps are intelligent guiding and asking only relevant inputs based on current context, situation or requirement. Makes app development more modular. The user has option to go back to previous input or any save point. Applications can be modelled using rule-based approach, the flow can be changed on-the-fly. Code is separate from execution engine (called as inference engine) thus makes it possible to modify the app's logic whenever needed.

How app logic is developed using expert system-based approach?

Rule-based apps implement *business logic* through set of rules. Rules may include complex conditions and functional code. Set of rules is called as *knowledge-base*. Let us start working on *Fruit Recognition App* which recognizes fruit based on the characteristics asked through Q&As to the end user. Consider the knowledge base consisting of following 17 rules. (Source: *Gonzalez, A. J., & Dankel, D. D. (1993); The engineering of knowledge-based systems, theory and practice. Englewood Cliffs, NJ: Prentice Hall*)

Rule 1:
If shape = long
and color = green
Then fruit = banana

Rule 1A:
If shape = long
and color = yellow
Then fruit = banana

Rule 2:
If shape = round
and diameter > 4 inches
Then fruitclass = vine

Rule 2A:
If shape = oblong
and diameter > 4 inches
Then fruitclass = vine

Rule 3:
If shape = round
and diameter < 4 inches
Then fruitclass = tree

Rule 4:
If seedcount = 1
Then seedclass = stonefruit

Rule 5:
If seedcount > 1
Then seedclass = multiple

Rule 6:
If fruitclass = vine
and color = green
Then fruit = watermelon

Rule 7:
If fruitclass = vine
and surface = smooth
and color = yellow
Then fruit = honeydew

Rule 8:
If fruitclass = vine
and surface = rough
and color = tan
Then fruit = cantaloupe

Rule 9:
If fruitclass = tree
and color = orange
and seedclass = stonefruit
Then fruit = apricot

Rule 10:
If fruitclass = tree
and color = orange
and seedclass = multiple
Then fruit = orange

Rule 11:
If fruitclass = tree
and color = red
and seedclass = stonefruit
Then fruit = cherry

Rule 12:
If fruitclass = tree
and color = orange
and seedclass = stonefruit
Then fruit = peach

Rule 13:
If fruitclass = tree
and color = red
and seedclass = multiple
Then fruit = apple

Rule 13A:
If fruitclass = tree
and color = yellow
and seedclass = multiple
Then fruit = apple

Rule 13B:
If fruitclass = tree
and color = green
and seedclass = multiple
Then fruit = apple
Else fruit = unknown

Typical rule has syntax

If Condition1
and Condition2
and ….
Then Conclusion1
Else Conclusion2

Rule 1:
If shape = long
and color = green
Then fruit = banana

If *shape of fruit is long*
and *color of fruit is green*
Then *fruit is banana*

Where condition can be simple expression or statement returning boolean result (is either true or false) in Variable = Value format (e.g. shape = long means *shape of fruit is long*) or complex one. This can be assignment statement: assigning value to a variable. The conclusion is typically simple statement of form: Variable = Value (e.g. fruit = banana means *fruit is banana*). Not all rules have conclusion as fruit but describe characteristics of fruit for example rules: 2, 2A and 3 determine what kind of fruit class the fruit belongs to, within that rules 2 and 2A determine whether fruit class is vine or not.

Rule 1:
If shape = long
and color = green
Then fruit = banana

Rule 1A:
If shape = long
and color = yellow
Then fruit = banana

Rule 2:
If shape = round
and diameter > 4 inches
Then fruitclass = vine

Rule 2A:
If shape = oblong
and diameter > 4 inches
Then fruitclass = vine

Rule 3:
If shape = round
and diameter < 4 inches
Then fruitclass = tree

Rule 4:
If seedcount = 1
Then seedclass = stonefruit

Rule 5:
If seedcount > 1
Then seedclass = multiple

Rule 6:
If fruitclass = vine
and color = green
Then fruit = watermelon

Rule 7:
If fruitclass = vine
and surface = smooth
and color = yellow
Then fruit = honeydew

Rule 8:
If fruitclass = vine
and surface = rough
and color = tan
Then fruit = cantaloupe

Rule 9:
If fruitclass = tree
and color = orange
and seedclass = stonefruit
Then fruit = apricot

Rule 10:
If fruitclass = tree
and color = orange
and seedclass = multiple
Then fruit = orange

Rule 11:
If fruitclass = tree
and color = red
and seedclass = stonefruit
Then fruit = cherry

Rule 12:
If fruitclass = tree
and color = orange
and seedclass = stonefruit
Then fruit = peach

Rule 13:
If fruitclass = tree
and color = red
and seedclass = multiple
Then fruit = apple

Rule 13A:
If fruitclass = tree
and color = yellow
and seedclass = multiple
Then fruit = apple

Rule 13B:
If fruitclass = tree
and color = green
and seedclass = multiple
Then fruit = apple
Else fruit = unknown

Similarly, rules 4 and 5 determine seed class of the fruit. While rest of the rules can be put in one block which determine fruit name. This makes it possible to divide the rules into *logical blocks* and *sub-blocks* where a block represents certain business logic to be executed whenever required e.g. Rules 2 and 2A represent sub-block which determines *fruit class as vine* and rule 3 represents the sub-block which determines *fruit class as tree*.

How app logic is executed?

To execute app, expert system uses *inference engine* which pickups appropriate rule(s) and executes them.

How inference engine knows which is the first rule to execute?

In goal driven system, the inference engine is told what the app wants to derive/infer/know finally, is called as the *goal* of the app. In this app, the goal is to recognize *fruit*. It can be specific goal, for example, to determine *Is fruit apple or not*?

Scenario 1: With goal to determine *fruit* (this is name of variable called *goal variable*).

The inference engine will start searching the *block* of rules which contain goal variable (*fruit*) in their conclusion parts. Following rules will be picked up by the inference engine for execution (called as *conflict set*) on first come first basis (this can vary and is based on *strategy* of inference engine, ideally general rules can be put on top). Conflict set:

Rule 1: If shape = long and color = green Then fruit = banana Rule 1A: If shape = long and color = yellow Then fruit = banana Rule 6: If fruitclass = vine and color = green Then fruit = watermelon Rule 7: If fruitclass = vine and surface = smooth and color = yellow Then fruit = honeydew Rule 8: If fruitclass = vine and surface = rough and color = tan Then fruit = cantaloupe	Rule 9: If fruitclass = tree and color = orange and seedclass = stonefruit Then fruit = apricot Rule 10: If fruitclass = tree and color = orange and seedclass = multiple Then fruit = orange Rule 11: If fruitclass = tree and color = red and seedclass = stonefruit Then fruit = cherry Rule 12: If fruitclass = tree and color = orange and seedclass = stonefruit Then fruit = peach	Rule 13: If fruitclass = tree and color = red and seedclass = multiple Then fruit = apple Rule 13A: If fruitclass = tree and color = yellow and seedclass = multiple Then fruit = apple Rule 13B: If fruitclass = tree and color = green and seedclass = multiple Then fruit = apple Else fruit = unknown

Inference engine will start executing first rule in conflict set. The first rule is

Rule 1:
If shape = long
and color = green
Then fruit = banana

It starts satisfying the conditions of the rules one by one.

The first condition is shape = long, it checks whether variable *shape* already has value assigned to it or not. If not, then it asks the user *'What is shape of the fruit?'* (assume this prompt is assigned to variable *shape*).

a> If user inputs or selects 'long'. The inference engine checks rules in conflict set, whether any rule has shape long in conditions part, it removes the rule(s) where *shape is not long*. It means conflict set has only two rules now where *shape is long*.

Rule 1:
If shape = long
and color = green
Then fruit = banana

Rule 1A:
If shape = long
and color = yellow
Then fruit = banana

It continues to execute Rule 1. It evaluates the first condition which is true. Then it tries to satisfy the second condition, will check whether *color* variable is having value or not. If not, it will ask *'What is color of fruit?'*. If user enters 'green', the inference engine will remove the rules from conflict set which does not have color green (rule 1A will be removed).

Rule 1:
If shape = long
and color = green
Then fruit = banana

The second condition will be true. Once both the conditions are satisfied, it will execute the conclusion part, means it will assign value *banana* to goal variable *fruit*. Once the conclusion is reached the inference engine will stop execution as goal is derived.

Rule 1:
If shape = long
and color = green
Then fruit = banana

In case user enters ‘yellow’ for variable color, first rule will be discarded and inference engine will pick up next rule in sequence i.e. Rule 1A whose’ both the conditions will be satisfied and fruit will be assigned *banana*.

OR

b> If user enters ‘round’. The first rule is discarded and the rule(s) where shape is not round will be removed (i.e. Rule 1 and Rule 1A).

New conflict set:

Rule 6: If fruitclass = vine and color = green Then fruit = watermelon Rule 7: If fruitclass = vine and surface = smooth and color = yellow Then fruit = honeydew Rule 8: If fruitclass = vine and surface = rough and color = tan Then fruit = cantaloupe	Rule 9: If fruitclass = tree and color = orange and seedclass = stonefruit Then fruit = apricot Rule 10: If fruitclass = tree and color = orange and seedclass = multiple Then fruit = orange Rule 11: If fruitclass = tree and color = red and seedclass = stonefruit Then fruit = cherry Rule 12: If fruitclass = tree and color = orange and seedclass = stonefruit Then fruit = peach	Rule 13: If fruitclass = tree and color = red and seedclass = multiple Then fruit = apple Rule 13A: If fruitclass = tree and color = yellow and seedclass = multiple Then fruit = apple Rule 13B: If fruitclass = tree and color = green and seedclass = multiple Then fruit = apple Else fruit = unknown

The inference engine will pick up the first rule in in conflict set, means rule no.6.

Rule 6:
If fruitclass = vine
and color = green
Then fruit = watermelon

Inference engine tries to satisfy conditions of rule 6. The first condition is fruitclass = vine. It realizes that fruitclass is determined from other rules and not be asked to the user. Such variable is called as *sub-goal* variable. It puts rule 6 on hold and looks for rules which determine fruitclass is vine. Following rules determine fruitclass vine.

Rule 2:
If shape = round
and diameter > 4 inches
Then fruitclass = vine

Rule 2A:
If shape = oblong
and diameter > 4 inches
Then fruitclass = vine

However, rule 2A has shape oblong hence discarded. So next rule inference engine will evaluate in conflict set is rule no. 2. The first condition of rule 2 will be satisfied because shape is round. The inference engine will ask value of *diameter*. If user enters more than 4 inches, then sub-goal is derived means fruitclass = vine. Inference engine will remove rule 2 from the conflict set as it is executed and *all the rules* where *fruitclass not vine*. Since rule 13B has else part the rule will remain in conflict set (since first condition will always fail because *fruitclass is not tree*).

New conflict set

Rule 6:
If fruitclass = vine
and color = green
Then fruit = watermelon

Rule 7:
If fruitclass = vine
and surface = smooth
and color = yellow
Then fruit = honeydew

Rule 8:
If fruitclass = vine
and surface = rough
and color = tan
Then fruit = cantaloupe

Rule 13B:
If fruitclass = tree
and color = green
and seedclass = multiple
Then fruit = apple
Else fruit = unknown

The next available rule is Rule 6, which was kept on hold (on top on conflict set).

Rule 6:
If fruitclass = vine
and color = green
Then fruit = watermelon

The first condition will be satisfied as fruitclass is vine. Inference engine will try to satisfy the second condition by asking color of fruit.

If user enters ‘green’ then conclusion part is evaluated as fruit = watermelon.

If the user selects other color than ‘green’ say ‘yellow’.

Inference engine would discard the rule 6 and all the rules where color is not yellow. In that case next rule will be rule 7 which has color ‘yellow’.

Rule 7:
If fruitclass = vine
and surface = smooth
and color = yellow
Then fruit = honeydew

Rule 7 first condition is satisfied, it will ask for surface of fruit, if surface value entered is *smooth* then inference engine will conclude fruit = honeydew

If the user selects other color than ‘green’, ‘yellow’ and ‘tan’. Then rules 6,7 and 8 all will fail. Inference engine will evaluate Rule 13B.

Rule 13B:
If fruitclass = tree
and color = green
and seedclass = multiple
Then fruit = apple
Else fruit = unknown

Inference engine will evaluate else part of the Rule 13B fruit = unknown because first condition will fail.

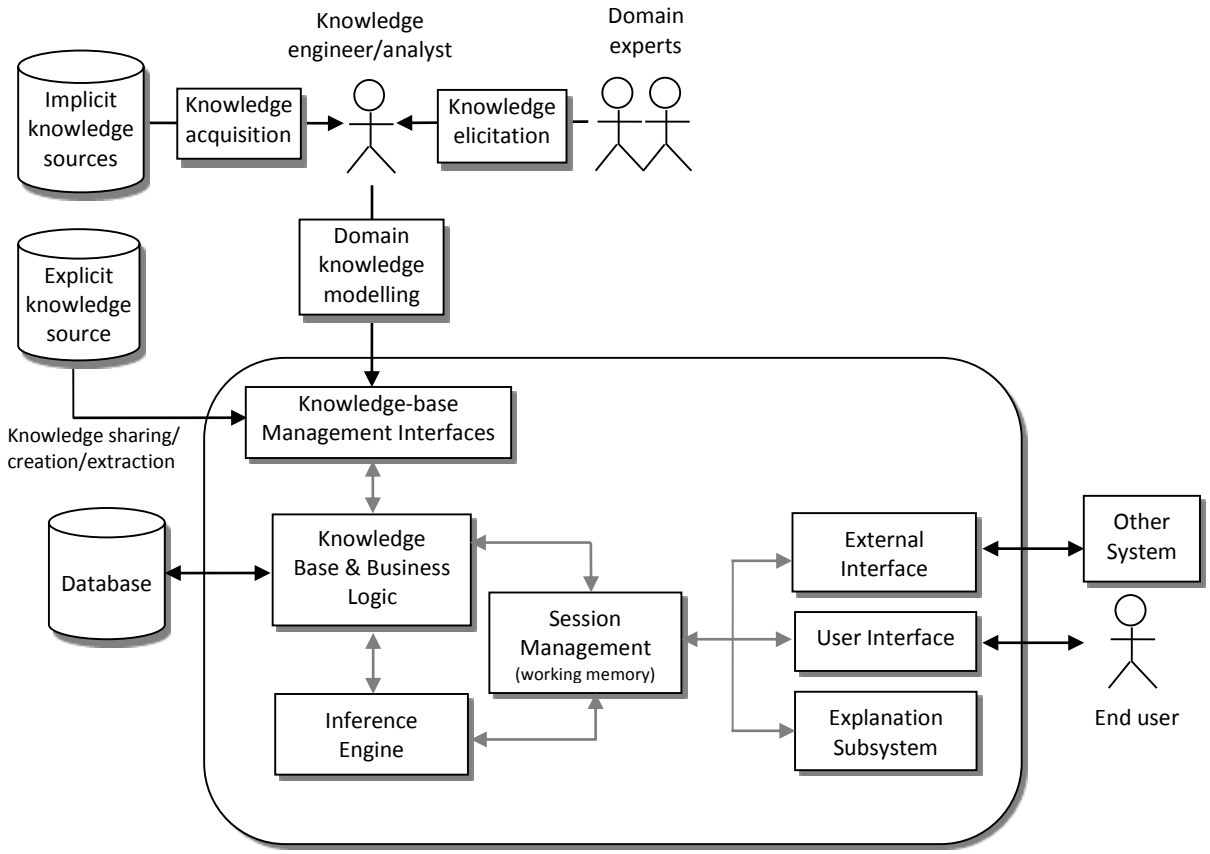
Scenario 2: Let us start with specific goal fruit = banana.

In specific goal, inference engine searches for rule(s) which have specific goal in conclusion part. In this example, inference engine searches *sub-block* of rules where conclusion is *fruit = banana*. It finds rules: 1 and 1A and puts them in conflict set. And starts executing first rule in the conflict set.

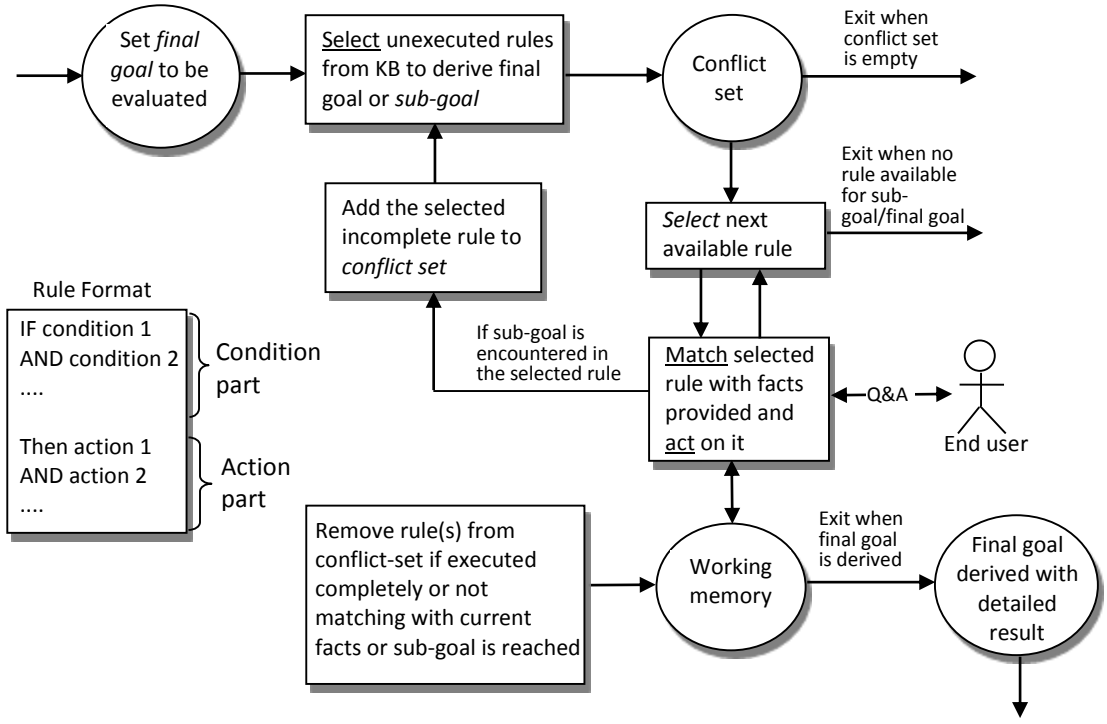
Rule 1:
If shape = long
and color = green
Then fruit = banana

Rule 1A:
If shape = long
and color = yellow
Then fruit = banana

If user inputs or selects 'long', it will ask color of fruit. If user selects green then first rule will be concluded (fruit = banana), if user selects color yellow, second rule 1A will be executed and will be concluded (fruit = banana). In case, user selects *shape* other than *long* or *color* other than *green* or *yellow*, inference engine will stop execution because it fails to infer *fruit is banana*.



Expert system components and players involved



Goal driven backward reasoning: Short Version

Goal driven backward reasoning: detailed