

Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 06
A First Course in Artificial Intelligence
Lecture – 75
Rule Based Expert Systems Conflict Resolution

(Refer Slide Time: 00:14)

Conflict Set

```
(p swapSort
  (array ^index <i> ^value <X>)
  (array ^index {<j> > <i>} ^value {<Y> <<X>})
  →
  (modify 1 ^value <Y>)
  (modify 2 ^value <X> )


(p ranking
  (next ^rank <r>)
  (totalMarks ^student <s> ^marks <m> ^rank nil)
  -(totalMarks ^marks > <m> ^rank nil)
  →
  (Modify 1 ^rank (<r> + 1)
  (Modify 2 ^rank <r>)
```

Working Memory

1. (array ^index 1 ^value 7)→
2. (array ^index 2 ^value 9)
3. (array ^index 3 ^value 3)
4. (array ^index 4 ^value 8)
5. (next ^rank 2)
6. (totalMarks ^student Rashmi ^marks 89 ^rank 1)
7. (totalMarks ^student Eva ^marks 79 ^rank nil)
8. (totalMarks ^student Anil ^marks 79 ^rank nil)
9. (totalMarks ^student Saifil ^marks 69 ^rank nil)

Conflict Set

{<swapSort 1 3>, <swapSort 2 3>, <swapSort 2 4>, <ranking 5 7>, <ranking 5 8>}



So, welcome back. So, we have been looking at this rule based production systems and we have seen in considerable detail the idea of matching of rules which data, rules have patterns, patterns may have variables, patterns may have Boolean test and data will have the data that these patterns must match and that is what is done by match algorithm.

What is the output of the match algorithm? It takes all the rules, matches it with all the data and gives you what is called as a conflict set. So, what is a conflict set? Let us look at this small example here. The two rules that we spoke about the first rule is for sorting we call it swap sort.

It simply says that if there are two elements which are out of place that, then we do not as far as the conflict set is concerned, actions do not concern us because we are simply trying to find out which rules are matching. But we know of course, that the actions were that we had to modify those two to records essentially.

If there is a record with value i and index i and value X and there is a later record with value less than X , then interchange the values of the two records. Likewise, we have seen the ranking rule, it says there is the next rank is r and there is the student s with marks m who has no rank assign and there is no student with higher rank, then assign the rank and implement the rank to next thing to this thing.

Let us say we have some working memory elements; we have got some four array elements here and we have some four students here with marks. One student has already been assigned a rank 1 and you will notice that the next rank is now 2 essentially.

What is the conflict set? The conflict said let us say these are the only two rules in our system and this is the only data, the conflicts that says which instances of these rules are matching and with what data essentially.

So, let us look at that. The first rule will match anything right, the first pattern of the first rule is match anything. It simply says there is an index i , it has a value X , we do not we have not stated what is I , we have not stated what is X , we have simply stated there is an index i and there is a value X stored in that.

So, obviously, it will match all the four elements, but we are interested in those matches which contribute to a rule matching which means that the second element also must matches

essentially. So, what is the second element here? It says that there is an index j which is greater than index i and it has a value Y which is less than the value X .

So, as you can see from this dash lines, this rule will match the 1st element and the 3rd element essentially. Why the 1st and the 3rd element? The 1st element has index 1 and value 7, the 3rd element has index 3 and a value 3 which is less than 7. So, that is why, it matches that.

So, this rule says that I have spotted two elements out of place, we can go ahead and swap them essentially, but there are other rules which are also waiting and that is why we have a conflict between them right and that is why we have a conflict set. Let us look at another example here.

Again, the next rank r will match this all the time and there are in fact, as you can see there are two students here, Eva and Anil both have got 79 marks, both of them do not have a rank assigned.

There is a student with higher marks 89, but the student already has the rank. Remember that this negative pattern said that the student must not have been assigned a rank. So, we can ignore Rashmi here, she is already got rank 1 and now there are two students Eva and Anil, both are candidates for the next rank which is 2 essentially.

What we have shown here is only this two lines which says that this rule matches 5 and 7 which means that this rule says gave Eva the rank 2 of course, there will be another instance of the rule which will say give Anil the rank 2.

So, if you draw a conflict set, if you look at the conflict set, then these two rules that we have shown as shown here in red. There are three instance of swap sort which have ready to swap elements, 1 and 3 which means 7 and; 7 and; 7 and 3 are out of place. So, the second rule says 2 and 3 which says that 9 and 3 are out of place and the third one says 2 and 4 which says that 9 and 8 are out of place.

So, there are three instances of rules which are ready to fire and they all three go into the conflict set. Likewise, there are two instances of ranking rule, we have already observed this both Eva and Anil can get rank 2 and so, therefore, we have these two elements in the rule and that is what is called as a conflict set.

So, a conflict set is a set each element is the tuple made up of rule name and the timestamps of the data with which it is matching. So, we will accept excess working memory elements by the timestamps essentially.

So, as you can see 1 and 3, 2 and 3, 2 and 4 for the swap sort this thing or 5 and 7, 5 and 8 for the ranking rules essentially, this is the conflict set. So, you can see that the task of match is to basically construct this conflict set. It look at all the rules, look at all the data and come up with this set out of which one will be selected.

(Refer Slide Time: 07:00)

Conflict Resolution

The following rule spots two values out of place...

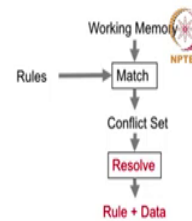
```
(p swapSort
  (array ^index <i> ^value <X>)
  (array ^index {<j> > <i>} ^value {<Y> <<X>})
→ .....
```

Given a large set of WMEs, the rule may match many pairs of elements!

The set of matching rules with data is called the conflict set (CS)

The Inference Engine selects one rule from the conflict set
- point to ponder – can we execute many rules in parallel?

Conflict resolution: choosing *which* rule to execute or fire next



Let us call conflict resolution. As seen on the right-hand side of the diagram, once you have had a conflict set, we give it to some resolved component and that will pick one rule with one data and we have already said that is the key to problem solving which rule to pick next.

This is a same swap short rule that we have been talking about. It basically spots two values which are out of place and says that it is time to swap them or something like that. Now, if there is a large set of working memory elements in the example, we saw they were only four out of which they were three which was ready to be swapped, the rule may match many pairs of the elements essentially and as we have said a set of that kind of matching rules with their data is called the conflict set.

And the inference engine selects one rule from the conflict set and says fire that rule or execute that rule, this is the term we use either fire or execute, but you think give some

thought to this matter. What would happen if you allowed multiple instances of rule to fire at the same time? What is the kind of difficulty we may face? Are there some problems which can be solved by parallel execution of rules or will that kind of some conflicting situations in some other situations?

We will not go in too much into that, but you can give some thought to that. We want to focus instead on conflict resolution which is choosing which rule to execute next or fire next essentially. So, conflict resolution ask this question which rule to pick essentially.

(Refer Slide Time: 09:07)

Conflict Resolution Strategies

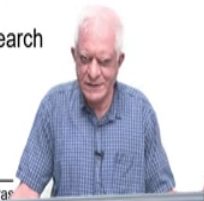
The key to effective problem solving is to make informed choices!

In heuristic search this was done by the heuristic function $h(N)$.

In rule based systems, the rules encapsulate such knowledge gleaned from human experts.

When more than one rule matches the data the inference engine has to make the choice of which rule instance to execute
- somewhat analogous to which candidate to inspect during search

We examine several Conflict Resolution Strategies



Let us look at that. As I said people will implement the rule-based systems for us to use or if you want to say people like us so, implement rule-based systems for some user to use have to offer a set of strategies which the user can select from. What are the different strategies that

one should consider? Let us quickly go over them and we will later go in more detail as to how to implement them.

As we said the key to effective problem solving is to make informed choices. In heuristic search, we try to use a heuristic function to help us decide which move to select next, but we also said that the heuristic function should be given to us by the user at least in this course as what we have said.

I must have mentioned during the planning lectures that the community has tried to evolve domain independent heuristics in the planning tasks which says basically that somehow if you can look at the problem and get a rough idea of which move will take shorter amount of time a rough idea not necessarily correct, then we can use that as a domain independent heuristic.

And the way that the planning community does that is by solving what is called as a relaxed version of a planning problem in which you relax the conditions when an operator applies and then, use that to solve the problem and typically those relax problems can be solved in polynomial time. So, that even the search space is exponential, the heuristic function is useful to have, such a heuristic function is that useful to have. Anyway we will not get back into that.

In rule-based systems, the rules encapsulate such knowledge gleaned from human experts. Somehow, we have said that the that of all the things that you can do the expert tells us as to which what are the things that can be done essentially.

The problem that we are now facing in conflict resolution is that when more than one rule matches data, the inference engine has to make the choice of which rule instance to execute essentially. This is somewhat analogous to in heuristic search which element to inspect next essentially. So, we look at various conflict resolution strategies now.

(Refer Slide Time: 11:57)

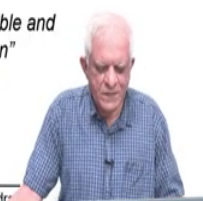
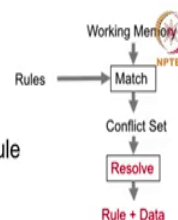
Conflict Resolution: Refractoriness

This says that a rule instance may fire only once with a set of matching WMEs. This is particularly relevant when the selected rule does not modify the WMEs matching its preconditions.
- else only this rule would keep firing

The idea of refractoriness comes from the way neurons fire in animal brains. When a neuron has received excitation that crosses its threshold, it fires once, and then waits for new signals to come in.

"the insensitivity to further immediate stimulation that develops in irritable and especially nervous tissue as a result of intense or prolonged stimulation"
- [Merriam Webster Dictionary](#)

Refractoriness happens naturally in the Rete Algorithm



The first strategy is called refractoriness. It says that a rule instance may fire only once with a set of matching working memory elements. This would be particularly relevant if the selected rule instance does not modify the working memory that was matching its preconditions. If it does not modify them, it will match again in the next cycle. So, we do not want to keep firing that same rule again and again and again.

So, this is a strategy which says that each rule with a given set of data and remember that the data is identified by their timestamps. Each rule with a given set of data will fire only once and not more than once and that strategy is called refractoriness. It is a key part of all systems that you want to implement.

The idea comes from the way neurons fire in animal brains. The way neurons fire is that they receive excitation from other neurons and when there is enough excitation built up in a

enough potential built up and goes beyond the threshold, then they fire in turn and that lose a stream of pulses down its axon essentially.

So, once it fires, the excitation is kind of finished and it does not fire again essentially and that is what is called refractoriness. The idea that we are using in rule-based systems is similar in nature also as you will see implemented in a very similar fashion.

Just a word from the Merriam Webster dictionary, the insensitivity to further immediate stimulation that develops in irritable and especially nervous tissue as a result of intense prolonged simulation.

So, you keep giving the same stimulus, do not expect the system to keep responding at the same piece essentially. As we will see this happens naturally in algorithm that we will study called the rete algorithm or the rete algorithm as some people call it.

(Refer Slide Time: 14:21)

Conflict Resolution: Lexical Order

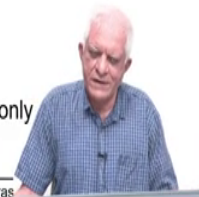
This says that of all the rules that have matching instances choose the first one that the user has stated.

And if a rule has multiple instances with different data, then choose the instance that matches the earlier data.

This strategy places the onus of this choice on the user. The user is more like a programmer.

This strategy is used in the programming language Prolog.

Prolog thus deviates from the idea of declarative programming envisaged by pure logic programming, in which the user would only state the relation between input and output.



The next strategy we can talk about is lexical order. This says that of all the rules that have matching instances choose the first one that the user has stated. So, presumably the user you can imagine a user sitting with pencil and writing down all the rules, take them in the lexical order in which the user has written the rules. So, clearly you can see that the onus is shifting to the user here.

If a rule has multiple instances with different data so, we have seen that in swap sort or ranking this thing that a rule may have multiple instances which have fired. Lexical strategy says choose the instance that matches the earlier data, the data that was first presented and we can do this by looking at the timestamps essentially.

You can see that the strategy places the onus of this choice from the user. The user is more like a programmer essentially. We will see next week sometimes is that this strategy is used

in the programming language Prolog which is why it is more of a programming language and less of an expert system shell that OPS5 is considered to be.

Because an OPS 5, these constraint on the user is not there to write the rules in a particular order which would make it more efficient or we should make even work essentially, you can write programs in prolog if you change the order, then it will just simply go into an infinite loop.

So, prolog in that sense deviates from the idea of pure declarative programming. So, in fact, some people differentiate between pure logic programming and actual logic programming which is like prolog essentially. In pure logic programming, you would only state the rules and if the solution was a consequence of the rules, it would be found, but in practice in prolog, you have to state the rules in a careful order.

(Refer Slide Time: 16:48)

Conflict Resolution: Specificity

This says that of all the rules that have matching instances choose the instance of the rule that is *most specific*.

Specificity can be measured in terms of the *number of tests* that patterns in rules need.

The intuition is that the more specific the conditions of a rule are, the more appropriate the rule is likely to be in the given situation

- remember that the Working Memory models the Short Term Memory of the problem solver
- rules constitute the problem solvers knowledge and reside in the quasistatic Long Term Memory

Specificity can facilitate default reasoning....



Coming to OPS 5, OPS5 has two very interesting conflict resolution strategies one of them is called specificity. This says that of all the rules that have matching instances, choose the instance of rule that is most specific essentially.

And if you think about this and it is a very insightful idea is that of the many things that you can do the most specific one or the one which matches the most specific conditions we are talking about the left-hand side, we are not talking about the right-hand side. The right-hand side may have any number of actions, we do not care.

The conflict resolution strategy does not look at the right-hand side. It simply says that these are the matching rules and these are the working memory elements that you are matching and that is a conflict set. It does not look at the right-hand side at all. So, when you say most specific, we mean the most specific left-hand side essentially. So, the left-hand side which has the most amount of conditions is probably the best rule to fire and that is what this strategy of specificity says.

And it can be measured in the number of tests that patterns in the rules need essentially. So, if there are many patterns with many tests, then it is more specific than a rule which has fewer patterns and fewer test. This strategy says if there are two such rules which are candidates in the conflict set, choose the one which is more specific, which has more tests essentially.

The intuition is that the more specific the conditions of a rule are the more appropriate the rule is likely to be in a given situation. Remember that the working memory element, working memory models the short-term memory of the problem solver. So, everything that problem solver knows about the problem, the data that it has acquired is there in the working memory. If there are rules which are looking at more of the data, they are likely to be more specific and perhaps according to this strategy more relevant essentially.

Rules only consider the problem solving knowledge and they reside in the quasi static long-term memory. So, when I say quasi static, I mean it is almost static that most implementations you input the rules and forget about it and then just keep using it to solve

different instances of problems. So, rules do not change, but what the strategy is saying is that more specific rules are better than less specific rules or more specific rules are better than more general rules.

This gives us, takes us to this idea of default reasoning. So, we are making like small forays into reasoning, it is not really the main source of this course and we have already reached the 10th week so, we are coming towards the end of the course but as I said all these things are you know tied up together with one with the other. So, let us look at this idea of default reasoning very briefly from a domain which is favorite of mine.

(Refer Slide Time: 20:06)

Default Rules: An example from bridge bidding

```

(p pass-rule
 (turn-to-bid ^player <x>)
 (next ^after <x> ^is <y>)
 →
 (make (bid ^player <x> ^bidName pass)
 (modify 1 ^player <y>))


(p 1N-opening
 (turn-to-bid ^player <x>) -
 (next ^after <x> ^is <y>) -
 (hand ^player <x> ^points <<15 16 17>> ^shape balanced)
 -(bid ^player {<y> <x> <x> ^round 1})
 →
 (make (bid ^player <x> ^round 1 ^type opening
 ^bidName notrump ^denomination 1)
 (modify 1 ^player <y>))

```

Default action: Pass
(unless you have something to communicate)

Specificity selects the more specific rule when both rules match

Holding 15-17 high card points with balanced shape, bid 1NT if no one else has bid anything



Exercise: Write rules to conclude that birds fly but not if they are penguins

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

And that is from the game of contract which I will highly recommend all of you to look at. It is something which will be with you the rest of your life once you have gotten into it anyway. So, here is an example from bridge bidding. So, the game has two parts, we did mention it

during game playing right. One is the first part is the auction where you bid for a contract and the second part is the actual playing out or trying to fulfill the contract. So, we are talking about bidding here and we are talking about default rules.

So, if you want to write a rule-based system, you must implement it in such a way that it is complete which means that in any situation, it tells you what to do. The idea of default reasoning is that if you cannot think of anything very specific, there must be some very general rules or default rules which will tell you that you can always do this thing essentially. So, the system should never get stuck essentially. So, that is easy to implement in bidding here.

There is a default action which says pass, do not make a bid or you know do not enter into the auction essentially and if we call this a pass rule, it simply says that if turn to bid of player x and so, just ignore the second pattern here for the moment. The first pattern simply says if it is your turn to bid, then make a pass bid ok. If you do not know anything else, just say pass. Pass basically means that I do I am not making a bid for the contract essentially.

The second pattern is more to do with implementing the game of the bid of the game of bridge. So, as you probably remember from the games lectures, it is a game played between four people so, you can see that there are four players typically we call them north, east, south, west and bidding goes or play also goes in this direction and the second pattern here in this rule, it says next after x is y. So, if x is playing, then after that y will play.

So, what does this? This means that in your knowledge base or database, they would be four records and they would say after north east plays, after east south plays, after south west plays and after west north plays. So, that is simply to take care of the implementing the game and essentially, in this says that if x has made a move which is to bid pass here, then the next player to bid is y and because that is what the record is saying that after x it is y. So, for example, after north it is east.

The important part is that we are talking about the default action that if you have nothing, no reason to bid something else just pass you know and I have friends whose status on their

social media site says pass is also a bid because there are people who do not know when to stop.

Here is this one rule which is more specific than the pass rule and we you do not want to get into the game here, but essentially there is something called you measure high card points in a game and you look at the shape of the hand, how many hearts you have, how many spades you have and things like that.

So, if you have 15 to 17 high card points and if your shape is balance, then bid 1 no trump if nobody else has been anything. So, this is what is sometimes called as the opening bid essentially. If nobody else has bid anything and if you have this pattern, then make this bid of 1 more trump and as you can see in of swipe like language, you can write this as a rule.

So, the first pattern is still the same, it is excess player to bid. The second pattern simply says after x, y will bid. The third patterns says this pattern that we are talking about that points are in the range 15 to 17.

So, the points are either 15 or 16 or 17 and the shape is balanced let us assume that you know the shape is a categorical attribute and you can have different names for different shapes and let us stick to some simple thing here and it says that no other player, no y which is not the same as x has made a bid in round 1. So, you are the first person to bid, then you can make this bid called notrump 1. So, 1 notrump essentially.

The purpose of giving this here is simply to point out the use of specificity as a strategy. If you have two rules which are in contention, choose the rule which has more specific. So, if you have something to bid in the case of bridge, make that bid.

If you have nothing to bid, you do not have a pattern which calls for a bid, then pass essentially. So, the rule specificity will make the right bid here. If you had used some other strategy, then who knows with a good hand also you might pass and we have to look for another partner or something.

Here is a small exercise write rules about birds essentially. So, you want to be able to say that birds can fly which is nice piece of knowledge to have, but sometimes they do not fly so, as if they are penguins they cannot fly, if they are ostriches, they cannot fly, if it is emu, you cannot fly or if it is dead, it cannot fly or if it is a caged bird you know you must have heard about that in India, then it cannot fly. Try to implement a set of rules which will conclude saying that it is a sparrow and a sparrow is a bird so, it can fly and things like that essentially.

(Refer Slide Time: 26:55)

Conflict Resolution: Recency

This says that of all the rules that have matching instances choose the instance that has the most recent WME.

Recency can be implemented by looking at the time stamps of the matching WMEs.

The intuition is that when a problem solver adds a new element to the Working Memory than any rule that matches that WME should get priority.

Recency can facilitate "a chain of thought" in reasoning.

The Conflict Set can be maintained as a priority queue.



One more strategy that we will look at is called recency. It says that of all the rules that have matching instances choose that instance that has the most recent working memory element. This is not looking at the rule, this is looking at the data. It is saying that the latest a rule which matches a latest data is the one that you should select essentially.

It can be implemented by looking at the time stamps of the matching working memory elements and I had mentioned that time stamps will be used in conflict resolution and this is where they are used here. The intuition is that when a problem solver adds a new elements to the working memory, then any rule that matches that working memory elements should get priority ok.

So, the idea is that recency can facilitate what can be thought of as a chain of thought, chain of thinking that you are making some inferences, you are making some actions and you want to you know carry forward on that recency will take care of that, you added a new element, it triggers a rule so, use that rule to match. We can maintain a priority queue; we will look at the implementation details as we go along essentially.

(Refer Slide Time: 28:21)

Conflict Resolution: MEA

The implementation of the language OPS5 from CMU has a strategy called MEA (Means Ends Analysis) based on the problem solving strategy espoused by Simon and Newell.

The idea is to partition the set of rules based on the *context* and focus on one partition at a time. One can think of each partition as solving a specific subgoal or reducing a specific difference.

The context is set by the *first pattern* in a rule. All rules in the same partition have the same first pattern.

The MEA strategy applies *Recency* to the first pattern in each rule, and *Specificity* for the remaining patterns.



So, a last strategy which has been used in OPS5 it is called MEA which stands for means ends analysis. You may remember this from the lectures we did on Simon and Newell's work and OPS5 has a specific way of looking at this strategy and it is as follows that the idea is.

So, you have rules which solve look at part of the problem and in the pattern directed fashion attempts addresses part of the problem. So, we saw this example where the state had many different patterns, and each pattern figured some rule and so on and so forth essentially.

What the means ends analysis strategy says that you partition the set of rules on the context and focus on one partition at the time. One can think of each partition is solving a specific sub goal or reducing a specific difference. But remember we are doing in a forward chaining match forward reasoning fashion not in backward reasoning fashion that we talked about earlier, but you can implement that aspect by creating these partitions.

So, it is like if you are building a house, you have to first make the basement, then you have to make the walls, then you have to make the windows and doors, then the roof and things like that. So, there are things to be done in a particular order and all the rules different kind of rules that you may use for making the basement will be put into one partition.

After you have built the basement, you will worry about the walls and so on. Of course, I am not saying that building a house is so similar, so simple, but you can think of it like that at least the operational part.

The context is set by the first pattern in a rule. So, in this MEA strategy which came with OPS5, the first pattern defined what is the context. All rules in the same partition had the same first pattern. So, you could when you are writing rules, you could say these are rules for constructing the floor, these are rules for constructing the house so, these are patterns which set the context essentially or constructing the walls and the constructing the windows and things like that.

What MEA strategy says is that you apply recency to the first pattern in the rule and if there is a tie, then specificity to the remaining patterns essentially. It is a very interesting strategy, and you should give some thought to what kind of problems this will be useful in essentially ok.

(Refer Slide Time: 31:23)



Towards an efficient implementation of a forward chaining rule based system



So, we will stop here. We have described the language of swipe, we have talked about it as a programming language, we have talked about it is an expert system shell, we have talked about the inference engine which said that there are there is a cycle of match, resolve and execute and keep doing this till somehow the problem tells you that it has been solved essentially.

Some termination criteria which will be implemented in some rule which will match at some point of time, but how to do this efficiently, it is a very key question and we will look at that in the next session. So, see you then.