32002: AI (Rule based system: Backward Chaining) **Spring 2014**

Lecture 14: February 03, 2014

Lecturer: K.R. Chowdhary *: Professor of CS (GF)*

14.1 Forward-Chaining

Complexity of Preconditions: The simplest method of evaluating preconditions is lookup in the knowledge-base. For example, in the case of query: "Is the pain intensification synchronous with respiration?", it is only necessary to check whether the value - "synchronous with respiration", is stored under the object - "pain intensification". With the logical connectives, *and*, *or*, *not*, this situation can be described. However, in the case of numerical or temporal relationships, the looking alone is not sufficient, and additional, calculations are required. For example, for the query "Did shortness of breath occur before throat pain", there is need of comparison, using the relations operators, like, less than, greater than, equal, which are needed to be applied along with the time information.

14.2 Backward Chaining

While forward chaining allows conclusions to be drawn only from a given knowledge base, a backward-chained rule interpreter is suitable for requesting still unknown facts. In a backward chaining system, you start with some hypothesis (goal) you are trying to prove, and keep looking for rules that would allow you to conclude that hypothesis, perhaps setting new sub-goals to prove as you go.

For this, the goal expression is initially placed in the working memory. Followed with, this the system performs two operations:

1. matches the rule 'consequent'with the goal,

2. selects the matched rule and places its premises in the working memory.

The second above corresponds to the transformation of the problem into subgoals. The process continues in the next iteration, with these premises becoming the new goals to match against the consequent of other rules. Thus, the backward chaining system, in the human sense are hypothesis testing.

Backward chaining uses stack. First, the goal is put in the stack, then all the rules which results to this goal are searched and put on the stack. These becomes the sub-goals, and the process goes on till all the facts are proved or are available as literals (ground clauses). Every time the goal and premises are decided, the unification is performed.

14.2.1 Algorithm

The algorithm for backward chaining returns the set of substitutions (*unifier*) which makes the goal true. These are initially empty. The input to the algorithm is knowledge-base (Γ), *goals*(α), and current substitution (θ), which is initially empty. The algorithm returns the substitution set (λ) for which is the goal is inferred.

Algorithm 1 backward-chaining(Input: Γ, α, θ) // α is a query, Γ is knowledge-base, θ current substitution (initially empty), λ represent substitution set (i.e., query result), initially empty.

1: $q^{'} \leftarrow \alpha\theta$
2: **for** each sentence $s \in \Gamma$, where $s = p_1 \wedge p_2 \wedge \ldots, \wedge p_n \rightarrow q$ and $\gamma \leftarrow unify(q, q^{'})$ and $\gamma \neq null$ **do**
3: $\alpha_{new} \leftarrow (p_1 \wedge p_2 \wedge \cdots \wedge p_n)$
4: $\theta \leftarrow \theta\gamma$
5: $\lambda \leftarrow$backward-chaining$(\Gamma, \alpha_{new}, \theta) \cup \lambda$
6: **end for**
7: return λ

Example 14.1 *Apply the backward-chaining algorithm for inferencing from a given knowledge-base.*

Solution. Let $\Gamma = \{man(x) \rightarrow mortal(x), man(socrates)\}$. Assume that it is required to infer "Who is mortal?" That is, goal $\alpha = mortal(w)$. Initially, θ is empty, hence, $q^{'} = \alpha\theta = mortal(w)$. From algorithm and knowledge-base Γ, $s = man(x) \rightarrow mortal(x)$. Thus, $\gamma = unify(man(x), man(w)) = \{w/x\}$. Also, the new goal, $\alpha_{new} \leftarrow man(x)$. The new value of current substitution is, $\theta \leftarrow \theta\gamma = \{w/x\}$. Next, compute $\lambda = backward - chaining(\Gamma, man(x), \{w/x\}) \cup \lambda$.

Second iteration: To compute the λ in second iteration, we apply the algorithm in a recursive mode, and get $q^{'} = man(x)\{w/x\} = man(w)$. Next, $\gamma = unify(man(socrates), man(w))$. This results to $\gamma = \{socrates/w\}$, and is an inference. □

14.2.2 Goal Determination

If the goal is not known in the knowledge base, the rule interpreter first decides whether it can be derived or must be requested. In the case of derivation of goal, all the rules which includes the goal in their action part are processed. These can be found efficiently if they are indexed according to their action parts.

Backward chaining therefore contains implicitly a dialogue control, whereby the order of the question put depends upon the order of the rules for deriving a parameter and upon the order of the propositions in the precondition of a rule. This dependency on the order reduces the modularity of the rule system. The efficiency of the backward-chained rule interpreter is determined by the formulation of the goal: the more precise the goal, the smaller is the search tree of rules to be checked and questions to be asked (for example, in medicine: "What is name of disease?"as against "Is X the name of disease?").

The examples of back-ward chained rule interpreter are MYCIN, and PROLOG.

14.3 Forward v/s Backward Chaining

Whenever the rules are such that typical set of facts relate to large number of conclusions, i.e., the *fan-out* is larger than *fan-in*, it is better to use backward chaining. On the other hand, when the rules are such

Table 14.1: Knowledge-base for Animal-Kingdom.

S. No.	Rule
1	$sponge(x) \rightarrow animal(x)$
2	$arthopod(x) \rightarrow animal(x)$
3	$vertebrate(x) \rightarrow animal(x)$
4	$fish(x) \rightarrow vertebrate(x)$
5	$mammal(x) \rightarrow vertebrate(x)$
6	$carnivore(x) \rightarrow mammal(x)$
7	$dog(x) \rightarrow carnivore(x)$
8	$cat(x) \rightarrow carnivore(x)$

that when a typical hypothesis can lead to many facts, i.e., *fan-in* is larger than *fan-out*, it is better to use forward chaining.

If there is a situation that all the required facts are known, and we want to get all the possible conclusions from these, the forward chaining is better. However, if the rules are such that facts are incompletely known, we cannot proceed from the facts and thus goal driven strategy should be used.

Forward chaining is often preferable in cases where there are many rules with the same conclusions, as shown in the following example.

Example 14.2 *Given a knowledge-base for an animal kingdom, infer* animal(Bruno) *after adding of* dog(Bruno).

The taxonomy of the animal kingdom includes such rules as shown in table 14.1:

In forward chaining, we will successively infer and add *carnivore(Bruno)*, *mammal(Bruno)*, *vertebrate(Bruno)*, and *animal(Bruno)*. The query will then succeed immediately. The total work is proportional to the height of the hierarchy of this taxonomy.

On the other way, if we use backward chaining, the query $\neg animal(Bruno)$ will unify with the first rule above, and generate the sub-query $\neg sponge\ (Bruno)$, which will initiate a search for $Bruno$ through all the subdivisions of sponges, and so on. Ultimately, it searches the entire taxonomy of animals looking for *Bruno*. □

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_4