CS323: AI (Rule Chaining) **Spring 2016**

Lecture 7: January 30, 2016

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

7.1 Forward Chaining

In a forward chaining system, you start with the initial facts, and keep using the rules to draw new conclusions (or take certain actions), given those facts. Forward chaining systems are primarily *data-driven*. Whenever an if pattern is observed to match an assertion, the antecedent is *satisfied*. When all the *if* patterns of a rule are satisfied, the rule is *triggered*. When a triggered rule establishes a new assertion or performs an action, it is *fired*. This procedure is repeated until no more rules are applicable (figure **??**).

The selection process is carried out in two steps:

1. *Pre-selection:* Determining the set of all the matching rules, also called the *conflict-set*.

2. *Selection:* Selection of a rule from the conflict set by means of a conflict resolving strategy.

7.2 Forward chaining Algorithm

A simple forward chaining algorithm 1 starts from the known facts in the knowledge-base, and triggers all the rules whose premises are the known facts, then adds the consequent of each into the knowledge-base. This process is repeated until the query is answered or until there is no conclusion generated to be added into the knowledge-base. We will use symbols θ, λ, γ to represent substitutions. The *unify* is a function unifies the newly generated assertion q' and the query α, and returns a unifying substitution λ if they are unified, else returns null.

Let α be the goal. The forward-chaining algorithm 1 picks up any sentence $s \in \Gamma$, where Γ is knowledge-base, and checks all possible substitutions θ for s. Let, the predicate form of s is $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q$. On substituting θ, say, it results to $(p_1 \wedge p_2 \wedge \ldots p_n)\theta = (p'_1 \wedge p'_2 \wedge \ldots p'_n)$, such that p'_i's $\in \Gamma$. Let $(p'_1 \wedge p'_2 \wedge \ldots p'_n) \rightarrow q'$, such that $q' = q\theta$. This q' is added into *new* inference. If the inference q' and goal α unify, then their unifier λ is returned as the solution, else the algorithm continues.

Algorithm 1 Forward-chaining(Input: Γ, α) // α is a query, Γ is knowledge-base

```
1: while True do
2:   new = {}
3:   for each sentence s ∈ Γ do
4:     Convert s into the format p1 ∧ p2 ∧ ··· ∧ pn → q
5:     for each substitution θ such that (p'1 ∧ p'2 ∧ ..., ∧ p'n) ← (p1 ∧ p2 ∧ ..., ∧ pn)θ for some p'is ∈ Γ do
6:       q' ← qθ
7:       new ← new ∪ {q'}
8:       λ ← unify(q', α)
9:       if λ is not null then
10:        return λ
11:      end if
12:    end for
13:    Γ ← Γ ∪ new // add new inferences in knowledge-base
14:  end for
15: end while
16: Return Fail
```

The algorithm may fail to terminate in the case when the raised query has no answer. For example, every natural number can be generated by recursively applying successor operation on a natural number, and assuming that 0 is natural number. This will lead to indefinite loop for very large numbers.

$$naturalnum(0).$$
$$\forall x[naturalnum(x) \rightarrow naturalnum(succ(x)).]$$

The following example demonstrates the manual run of forward chaining algorithm.

Example 7.1 *Produce the inference, given the knowledge-base* $\Gamma = \{man(x) \rightarrow mortal(x), man(socrates)\}$ *and query* $\alpha = mortal(w)$*, i.e., "Who is mortal?"*

Solution. We follow the algorithm 1 manually and note that $p_1 \wedge \cdots \wedge p_n = p_1 = man(x)$; $p'_1 = man(socrates)$, substitution $\theta = \{socrates/x\}$, and $q = mortal(x)$. Also,

$$\begin{aligned} q' &= q\theta \\ &= mortal(x)\{socrates/x\} \\ &= mortal(socrates). \end{aligned}$$

Also, $new = \{\} \cup \{q'\} = mortal(socrates)$.

On unification of α (i.e, $mortal(w)$), and q', the unifier λ obtained is,

$$\begin{aligned} \lambda &= unify(q', \alpha) \\ &= unify(mortal(socrates), mortal(w)) \\ &= \{socrates/w\}. \end{aligned}$$

Hence, the answer for query is $w = socrates$. □

7.3 Backward Chaining 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 (initially empty). The algorithm returns the substitution set λ for which the goal is inferred. The algorithm 2 is the backward-chaining algorithm.

Algorithm 2 Backward-chaining(Input: Γ, α, θ) // α is a query, Γ is knowledge-base, θ current substitution (initially empty), λ represent substitution set for the query to be satisfied (initially empty).

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

Example 7.2 *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 answer for "Who is mortal?" That is, goal $\alpha = mortal(w)$, find w. The loop iterations in the backward-chaining algorithm 2 are as follows:

1st Iteration: Initially θ is empty, hence, $q' = \alpha\theta = mortal(w)$. From algorithm and knowledge-base Γ, the sentence is, $s = (man(x) \rightarrow mortal(x))$. Next, $\gamma = unify(mortal(x), mortal(w)) = \{w/x\}$. Also, the new goal, $\alpha_{new} = man(x)$. The new value of current substitution is, $\theta \leftarrow \theta\gamma = \{\}\{w/x\} = \{w/x\}$. Next, compute $\lambda = backward\text{-}chaining(\Gamma, man(x), \{w/x\}) \cup \lambda$, as a recursive call.

Recursive call: We apply the algorithm in a recursive mode, and get $q' = man(x)\{w/x\} = man(w)$. Next, the subgoal $man(w)$ (i.e., q') matches with other subgoal (it is a *fact*) $man(socrates) \in \Gamma$. Hence, $\gamma = unify(man(socrates), man(w)) = \{socrates/w\}$. Next, the new goal is, $\alpha_{new} = man(socrates)$ and new substitution is:

$$\begin{aligned}\theta &= \theta\gamma \\ &= \{w/x\}\{socrtaes/w\} \\ &= \{socrates/x\}.\end{aligned}$$

In the next call of recursion at step 5, $q' = \alpha\theta = man(socrates)$ $\{socrates/x\}$, the substitution fails, as there is no x where "socrates" can be substituted. Hence, $q' = null$. Since γ is null, the *for loop* at step 3 terminates, and returned value of λ is $\{socrates/w\}$, i.e., $w = socrates$, or $mortal(socrtaes)$.

□