Forward Chaining vs. Backward Chaining

Knut Hinkelmann
Forward Chaining vs. Backward Chaining

Logical Rules can be applied in two directions

- **Backward chaining**
  - start with the desired conclusion(s)
  - work backwards to find supporting facts
  - corresponds to modus tolens
  - **goal-directed**

- **Forward chaining**
  - Starts from the facts
  - apply rules to find all possible conclusions
  - corresponds to modus ponens
  - **data driven**
Example of a Declarative Knowledge Base

Father(peter, mary)
Father(peter, john)
Mother(mary, mark)
Mother(jane, mary)

Father(X, Y) AND Father(Y, Z) → Grandfather(X, Z)
Father(X, Y) AND Mother(Y, Z) → Grandfather(X, Z)
Mother(X, Y) AND Father(Y, Z) → Grandmother(X, Z)
Mother(X, Y) AND Mother(Y, Z) → Grandmother(X, Z)
Father(X, Y) AND Father(X, Z) → Sibling(Y, Z)
Mother(X, Y) AND Mother(X, Z) → Sibling(Y, Z)

The rules can be used to
• Derive all grandparent and sibling relationships (forward chaining)
• Answer questions about relationships (backward chaining)
Illustrating Backward Chaining

Illustration Forward Chaining

Goal state: Z
Termination condition: stop if Z is derived or no further rule can be applied

Forward Chaining: Deriving ground Facts

- Usually for forward chaining the facts are ground, i.e. they do not contain variables.
- To ensure that the derived facts are ground, all the variables which occur in the consequence of the rule must occur in the antecedents of the rule.
- Unification is thus restricted to matching (one of the expressions is ground):
  - The condition can contain variables.
  - The matching fact does not contain variables.
Forward Chaining Procedure: Recognise – Select – Act Cycle

Let the fact base consist of facts $FB = \{F_1, \ldots, F_n\}$

1. **Recognise**: Match the conditions of the rules against the facts of the fact base, i.e. find all rules $C_1$ and $C_2$ and ... and $C_m \rightarrow H$ such that the conditions $C_1$, $C_2$, ..., $C_m$ can be unified with facts $F_1$, $F_2$, ..., $F_m$ with unifier $\sigma$ (the set of applicable rules is called conflict set)

2. **Select**: If there is more than one rule that can be applied, choose one to apply. Stop if no rule is applicable

3. **Act**: Apply the chosen rule by adding $H\sigma$ to the fact base, i.e. $FB = FB \cup \{H\sigma\}$

4. Stop if termination condition holds, otherwise and go to 1
Forward Chaining Strategies

- Forward chaining computes all the facts that can be derived from the knowledge base.

- Forward chaining strategies differ in step “Select”. Here are some examples of strategies:
  - Apply the rules sequentially
  - Randomly select a rule
  - Apply more specific rules first
  - Prefer rules where conditions match a recently derived fact
  - Derive consequences of a set of starting facts: Only apply rules where at least one condition matches either with a starting fact or a derived fact
    - Fact base contains facts that are generally true, e.g. insurance product
    - Starting facts describe a concrete situation, e.g. customer data
Choosing Forward or Backward Chaining

- **Backward Chaining**
  - If you already know what you are looking for

- **Forward Chaining**
  - If you don't necessarily know the final state of your solution
Decision Criteria for Forward or Backward Reasoning

- More possible goal states or start states?
  - Move from smaller set of states to the larger

- Is Justification of Reasoning required?
  - Prefer direction that corresponds more closely to the way users think

- What kind of events triggers problem-solving?
  - If it is arrival of a new fact, forward chaining makes sense.
  - If it is a query to which a response is required, backward chaining is more natural.

- In which direction is branching factor greatest?
  - Go in direction with lower branching factor

Branching Factor

Backward chaining more appropriate

Forward chaining more appropriate
Combining Forward Chaining and Backward Chaining in VisiRule: Statement Box

- The function of a statement box is to calculate a value from information that is already known.

- Statement boxes have three elements:
  - an editable **name** (balance_plus_order in example below)
  - an editable local **variable** (X in example below)
  - a statement assigning a value to the variable using the operator „is“ (X is balance + order_total.)

(Note: On the right of „is“ there is editable Prolog code which is used to calculate the value)
Statement Box with an Arithmetic Expression

- **Start1**
  - **Order_total**
    - > 500: Check cred
    - <= 500: Rule2

- **Rule2**
  - > 600: Check cred
  - <= 600: Balance

- **Balance**
  - balance_plus_order
  - X is balance + order_total
    - > 600: Check cred
    - <= 600: Rule3
Example: Calculating Leap Years