Rule-Based (Expert) Systems

Sven Koenig, USC

Russell and Norvig, 3rd Edition, Sections 9.3 and 9.4

These slides are new and can contain mistakes and typos.
Please report them to Sven (skoenig@usc.edu).

Example: Taxonomic Knowledge

- "All office machines get their energy from wall outlets."
- "All printers are office machines."
- "All laser printers are printers."
- "Hobbes is a laser printer."

Example: Taxonomic Knowledge

- Knowledge base in first-order logic
  - FORALL x IsOfficeMachine(x) IMPLIES HasEnergySource(x, WallOutlet)
  - FORALL x IsPrinter(x) IMPLIES IsOfficeMachine(x)
  - FORALL x IsLaserPrinter(x) IMPLIES IsPrinter(x)
  - IsLaserPrinter(Hobbes)
- We can use resolution to show that the knowledge base entails
  - HasEnergySource(Hobbes, WallOutlet)
- But the knowledge base and resolution are difficult to understand by non-experts and resolution is often slow (and non-trivial to implement), so we are looking for alternative ways to represent knowledge and reason with it.

Modus Ponens

$$\frac{P,\ P \text{ IMPLIES } Q}{Q} \qquad \frac{P(A),\ \text{FORALL } x\ (P(x) \text{ IMPLIES } Q(x))}{Q(A)}$$

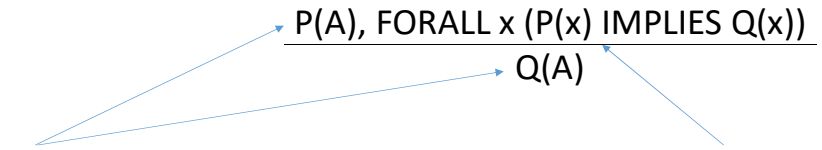
(write: "$\vdash^{\text{Modus Ponens}}$")

- Using $KB \vdash^{\text{Modus Ponens}} S$ to show $KB \models S$ is sound but not complete.

- Example:
  P IMPLIES Q, NOT P IMPLIES Q $\models$ Q
  but neither P IMPLIES Q, NOT P IMPLIES Q $\vdash^{\text{Modus Ponens}}$ Q
  nor P IMPLIES Q, NOT P IMPLIES Q, NOT Q $\vdash^{\text{Modus Ponens}}$ FALSE

Modus Ponens

- Modus Ponens uses a fact "P(A)" and a rule "if P(x) then Q(x)" to produce a new fact "Q(A)".



Fact
- Fact about a patient
- Knowledge the adventurer has about the cave system in the Wumpus World

If-then rule
- Medical knowledge
- Rules of the Wumpus World

Rule-Based Systems (= Production Systems)

- Knowledge base split into rule and working memory
- Rule memory (contains rules, does not change during execution):
  if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then IsOfficeMachine(x)
  if IsLaserPrinter(x) then IsPrinter(x)

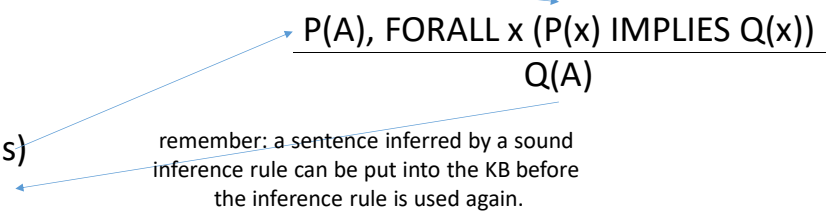
- Working memory (contains facts, changes during execution):
  IsLaserPrinter(Hobbes)

Forward Chaining

- Forward chaining is data-driven
- When a new fact p is added to working memory
  for each rule R such that p unifies with a premise of R
  if the other premises of R are known
  then add the conclusion of R to working memory and repeat

Forward Chaining

- Rule memory:
  if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then IsOfficeMachine(x)
  if IsLaserPrinter(x) then IsPrinter(x)
- Working memory:
  IsLaserPrinter(Hobbes)
  IsPrinter(Hobbes)



Forward Chaining

- Rule memory:
  if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then IsOfficeMachine(x)
  if IsLaserPrinter(x) then IsPrinter(x)

- Working memory:
  IsLaserPrinter(Hobbes)
  IsPrinter(Hobbes)
  IsOfficeMachine (Hobbes)
  HasEnergySource(Hobbes, WallOutlet)

Forward Chaining

- Rule memory:
  if IsOfficeMachine(x) then add HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then add IsOfficeMachine(x)
  if IsLaserPrinter(x) then add IsPrinter(x)

Could use actions other than “add (to working memory)” here, such as “delete from working memory” or “print”

- Working memory:
  IsLaserPrinter(Hobbes)

Forward Chaining

- Match phase (= find all applicable rules/unification binding combinations)
- Conflict resolution phases (= choose one rule/unification binding)
  - Don't fire (= use) a rule again with the same unification bindings
  - Use more recent facts from working memory
  - Use more specific rules
    if Mammal(x) then add Legs(x,4)
    if Mammal (x) and Human(x) then add Legs(x,2)
  - Use rules of higher given priority
    if ControlPanel(x) and Dusty(x) then execute Dust(x)
    if ControlPanel(x) and WarningLightOn(x) then execute Evacuate
- Act phase (= execute the conclusion of the chosen rule/unification binding)

Backward Chaining

- Backward chaining is query-driven (= hypothesis-driven)
- When a new query q is asked
  if a fact q' is in working memory that unifies with q
  then return the unifier of q and q'
  else
    for each rule R such that q unifies with the conclusion of R
      pose each premise of R as new query and repeat

Backward Chaining

- Rule memory:
  if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then IsOfficeMachine(x)
  if IsLaserPrinter(x) then IsPrinter(x)

$$\frac{P(A),\ \text{FORALL } x\ (P(x)\ \text{IMPLIES}\ Q(x))}{Q(A)}$$

- Working memory:
  IsLaserPrinter(Hobbes)

Query: HasEnergySource(Hobbes, WallOutlet)?
New Query: IsOfficeMachine(Hobbes)?

Backward Chaining

- Rule memory:
  if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  if IsPrinter(x) then IsOfficeMachine(x)
  if IsLaserPrinter(x) then IsPrinter(x)

- Working memory:
  IsLaserPrinter(Hobbes)

Query: HasEnergySource(Hobbes, WallOutlet)?
New Query: IsOfficeMachine(Hobbes)?
New Query: IsPrinter(Hobbes)?
New Query: IsLaserPrinter(Hobbes)?
IsLaserPrinter(Hobbes) holds because it is in working memory.

Rule-Based Systems (= Production Systems)

- Forward chaining
  good for design (for example, configuration planning)
  good for some diagnosis (find out everything we can from symptoms)

- Backward chaining
  good for some diagnosis (confirm hypothesis about diagnosis)

Rule-Based Systems (= Production Systems)

- Modularity
- Control isolated from knowledge base
- Easy Modification
- Explanation capability for its conclusions

Rule-Based Systems (= Production Systems)

- Conclusions are often not certain
  - if IsOfficeMachine(x) then HasEnergySource(x, WallOutlet)
  - If IsOfficeMachine(x) then it is highly likely that HasEnergySource(x, WallOutlet)
- We discuss probabilistic expert systems later to address this issue.