Models for Inexact Reasoning

Overview of Rule-Based Systems

Miguel García Remesal
Department of Artificial Intelligence
mgremesal@fi.upm.es
Rules and ‘Productions’

• ‘Production’ – synonym for rule

• English regular verbs
  – regular_verb + (past) → verb + ‘ed’

• Pig-Latin productions
  – consonant string → string + consonant + ‘ay’
  – vowel + string → string + ay
Work through this production system to decode the result

- \( z \rightarrow l \)
  - \( u \rightarrow p \)
  - \( p \rightarrow h \)
  - \( lw \rightarrow llo \)
  - \( q \rightarrow r \)
  - \( v \rightarrow w \)
  - \( r \rightarrow e \)

- \( uqzv \rightarrow ?? \)
Rules

• **Form**
  – IF antecedent THEN conclusion
  – IF condition THEN action
  – IF antecedent THEN goal

• **Interpreters**
  – Backward chaining
    • Trigger on conclusion/goal
  – Forward chaining
    • Trigger on antecedent/condition
Forward and Backward Chaining

• Rules
  – r1: IF may_rain THEN should_take_umbrella
  – r2: IF cloudy THEN may_rain

• Questions
  – “Should I take an umbrella?”
  – “What should I do if it is cloudy?”

• How did you answer the questions?
  – Which part of the rule did you look for? (‘match’)
Backward Chaining

• Rules
  – R1: IF may_rain THEN should_take_umbrella
  – R2: IF cloudy THEN may_rain

• “Should I take an umbrella?”
  – “Do the rules indicate I should take an umbrella?
    • Is there a rule about “taking umbrellas”?
      – R1: goal: should_take_umbrella
    • How can I prove that goal?
      – What has to be true for r1 to hold?
        » may_rain is the antecedent of r1
    • Can I prove that it may_rain?
      – R2: goal: may_rain
    • How can I prove that goal2
      – What has to be true for r2 to hold
        » cloudy is the antecedent of r2
    • How can I prove ‘cloudy’?
Backward Chaining

Goal_1 $\rightarrow$ Goal_2

Goal_2 $\rightarrow$ Goal_3

Goal_3 $\rightarrow$ Goal_4

Goal_4 $\rightarrow$ Goal_5

Question

rules
Backward Chaining

• Rules
  – R1: IF may_rain THEN should_take_umbrella
  – R2: IF cloudy THEN may_rain
  – R3: IF may_be_intense_sun THEN should_take_umbrella
  – R4: IF summer AND in_tropics THEN may_be_intense_sun

• “Should I take an umbrella?”
  – “Do the rules indicate I should take an umbrella?
    • Is there are rule about “taking umbrellas”?  
      – R1: goal: should_take_umbrella
    • What is antecedent for r1?  
      – R1: antecedent may_rain
    • Can I prove that it may_rain?  
      – R2: goal: may_rain
    • How can I prove may_rain  
      – R2: antecedent: cloudy
    • BUT NOT CLOUDY!
Backward Chaining:  *Backtracking*

**Rules**

- R1: IF may_rain THEN should_take_umbrella
- R2: IF cloudy THEN may_rain
- R3: IF may_be_intense_sun THEN should_take_umbrella
- R4: IF summer AND in_tropics THEN may_be_intense_sun

**“Should I take an umbrella?”**

- Are there any other rules about umbrellas?
  - R3: goal: should_take_umbrella
- What is antecedent of R3?
  - R3: antecedent: summer AND in_tropics
Backwards Chaining with Backtracking

Goal_1 $\rightarrow$ Goal_2

Goal_2 $\rightarrow$ Goal_3

Goal_3 $\rightarrow$ Goal_4

Goal_4 $\rightarrow$ Goal_5

Goal_5 $\rightarrow$ Goal_6

Goal_6 $\rightarrow$ Goal_4

Goal_7 $\rightarrow$ Goal_6

Goal_8 $\rightarrow$ Goal_7

rules

Direction of reasoning

Question
Backwards Chaining Systems

• **MYCIN**
  – ‘The original expert system’
    • Diagnosis of acute infections (Meningitis, blood infections)
      – Still a good example of how it works
    • Also used
      – uncertain reasoning
      – Explanation
        » ‘How’ did you prove that?
        » ‘Why’ are you asking me that?
  • Never used ‘for real’

• **PROLOG**
  – One of the two standard AI languages
    • A simple backwards chaining engine with backtracking
Backwards Chaining Engines usually written ‘backwards’

• **Goal ← Antecedent**
  – Umbrella ← may_rain
  – may_rain ← cloudy

• **Prolog ‘Edinburgh’ notation**
  – umbrella :- may_rain.
  – may_rain :- cloudy.

NB upper and lower case very important in Prolog
Forward Chaining

• Rules
  – R1: IF may_rain THEN should_take_umbrella
  – R2: IF cloudy THEN may_rain

• “What should I do if it is cloudy?”
  – “What do the rules indicate I should do if it is cloudy?”
    • Is there a rule that applies when it is cloudy?
      – R2: antecedent: cloudy
    • What do I conclude from that antecedent, ‘cloudy’
      – R2: conclusion: may_rain
    • Is there a rule that applies when it may_rain?
      – R1: antecedent: may_rain
    • What do I conclude from that antecedent: ‘may_rain’
      – R1: conclusion: should_take_umbrella
Forward chaining

‘Production Systems’

- Vocabulary used differently on west and east coast of US for many years
  - On east coast, ‘production systems’ means forward chaining
  - On west coast, ‘production systems’ just means rule based systems
  - Usually, and in this course, ‘Production System’ means ‘forward chaining’
Forward Chaining

Fact_1

Fact_1 → Fact_2

Fact_2 → Fact_3

Fact_3 → Fact_4

Fact_4 → Fact_5

Action=Fact_5

rules

Direction of reasoning
Production system interpreter

• Objectives:
  – Fire rules as the facts come in to the knowledge base
  – Never fire a rule unless its conditions are satisfied
  – Fire every rule whose conditions are satisfied

• Are these objectives consistent.
  – Forward chaining rules sometimes called ‘demons’
    • From a system called “Pandemonium”
  – How can they be made consistent?
Production System Strategy

• All rules tested at each cycle
• Only one rule fires at a time
Production System Cycle

1. Test all rules
2. Put all rules satisfied into the ‘conflict set’
3. Choose one rule from the conflict set
4. Fire the rule
5. Update the dynamic database
6. Repeat until goal reached or no more rules satisfied
Conflict Resolution

R1: IF sky=cloudy THEN expect=rain
R2: IF expect=?X THEN weather=?X
R3: IF sky=cloudy AND temperature=freezing
    THEN expect=snow
R4: IF weather=rain THEN temperature=above_freezing

What happens if ‘sky=cloudy’?
What happens if ‘sky=cloudy and 
    ‘temperature=freezing’?
Possible Conflict Resolution Strategies

• Specificity
• Priority
• Lexical Ordering
• Source file ordering
• Explicit rules for conflict resolution
  – a rule based system within a rule based system
Basic Production System Architecture

Dynamic Memory
- execute
- tickle

Rule Execution
- Select (resolve conflicts)

Conflict Set
- check satisfaction

Rule Store