UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Artificial Intelligence

Expert Systems

LESSON 17

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Introduction

- One of the main applications of knowledge-based systems in Artificial Intelligence
  - Encoding human experts' problem-solving knowledge in specific application domains for which no single algorithmic solution exists (e.g., medical diagnosis)
  - Commonly used as decision support systems, i.e., to support (not to replace) expert's decisions
  - Problem-independent architecture for knowledge representation and reasoning
  - Knowledge representation
    - IF … THEN … rules

Definitions: Expert System

- Expert System (ES), aka
  - Knowledge-based system (KBS)
  - Knowledge-based decision support system (KBDSS)
  - Intelligent Decision Support Systems (IDSS)
  - Rule-based system (RBS)
  - Production system (PS)
- *A computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert* [in a limited domain] – Feigenbaum
- *A computer system that operates by applying an inference mechanism to a body of specialist expertise represented in the form of knowledge*
- *A program intended to make reasoned judgments or give assistance in a complex area in which human skills are fallible or scarce*
- *A program designed to solve problems at a level comparable to that of a human expert in a given domain*
- A system embodying specialist expertise (e.g., medical knowledge)

Motivations

- Mainly
  - Limitation of general problem-solving approaches pursued in AI until the 1960s
  - First expert systems: 1970s
  - Widespread use in the 1980s
    - Many commercial applications
    - Used in niche/focused domains since 1990s

Main Current Applications of Expert Systems

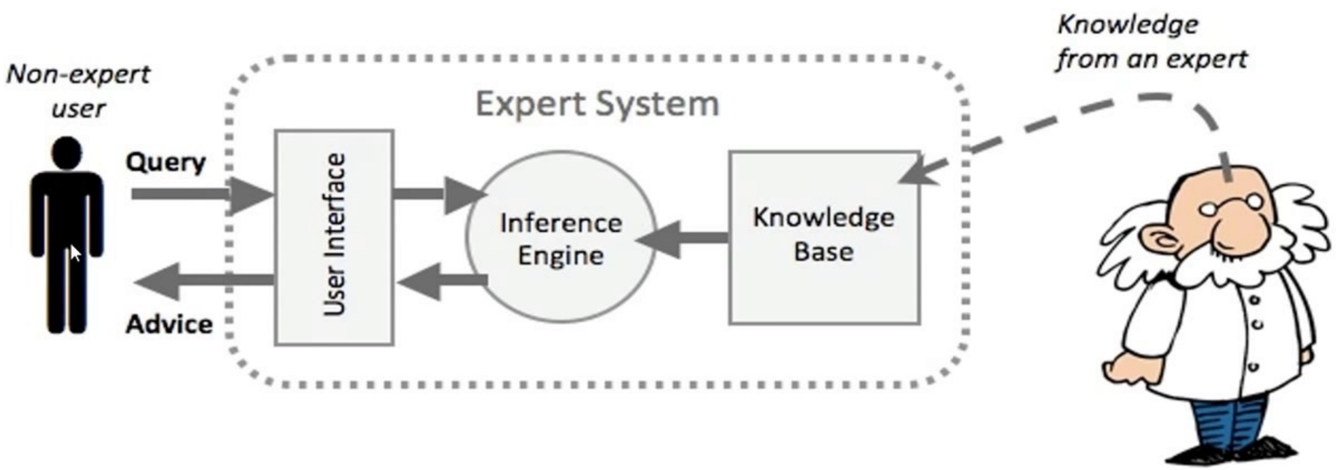
- Medical diagnoses
- Geology, botany (e.g., rock and plant classification)
- Help desk
- Finance
- Military strategies
- Software engineering (e.g., design patterns)

General ES Characteristics

- Symbolic reasoning
  - Knowledge must be represented symbolically
    - Various representations possible
      - Rules, trees, frames, description logics
- Deal with subjects of considerably complexity
  - Typically in a narrow, specialized domain
  - Expert-level solutions to complex problems
- Knowledge separated from processing (i.e., inference)
  - Reusable shell
- High-quality performance
  - Speed does not mean much if the answer is wrong
- Must be fast
  - A correct answer does not mean much if it comes too late
  - E.g., the patient has died or nuclear power plant has exploded
- Capable of explaining and justifying solutions
  - Interpretable/understandable
- Flexible
  - New information incorporated, old information fixed/corrected

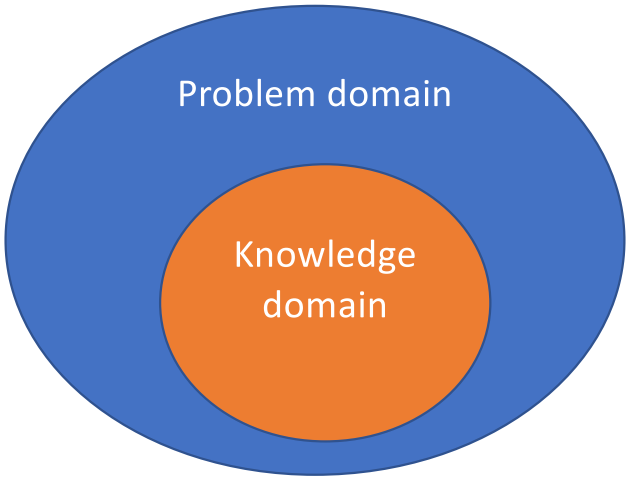
High level Schematic of ES

- Purpose of ES
  - Replace (i.e., automate) or provide support for human experts in solving or advising on real-world problems



Problem vs Knowledge Domain

- An expert's knowledge is specific to one problem domain, e.g., medicine, finance science, engineering, etc.
- The expert's knowledge about solving specific problems is called the knowledge domain
- The problem domain is always a superset of the knowledge domain

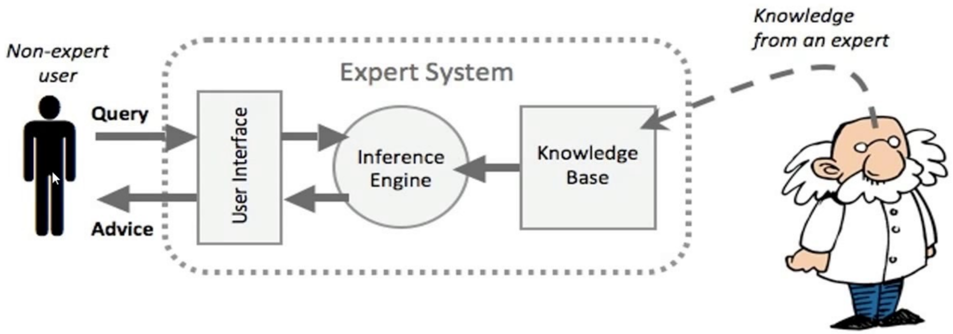


Designing the Knowledge Base of Expert Systems

- Two main distinct roles
  - Knowledge engineer
  - Domain expert
- Main issues
  - Defining suitable data structures for representing facts (specific problem instances) in working memory
  - Suitably eliciting expert's knowledge (general knowledge) and encoding it as IF … THEN rules in rule memory

Key Components of ES

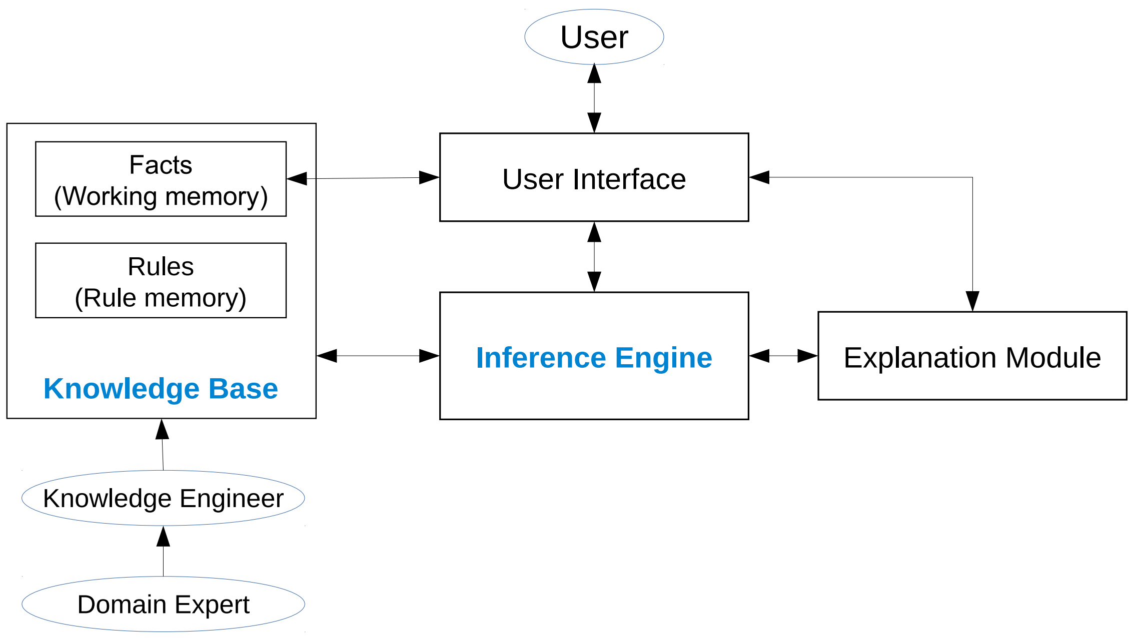
- Knowledge base
  - Container of symbolically and formally represented knowledge
  - Many representations possible
    - Rules, tree, frames
    - Semantic networks, Bayesian networks
- Inference Engine
  - Reasoning
    - Draws conclusions from knowledge base
    - Reasoning/inference approach must pair with representation



Additional Components of ES

- User Interface
  - The mechanism by which the user and system communicate
- Knowledge base editor
  - Aka, Knowledge base acquisition facility
  - An automatic way for the user to enter knowledge into the system bypassing the need for explicit coding expertise
- Explanation system (Justifier)
  - Explains the reasoning of the expert system to the user

Expert System Architecture



Expert Systems: Rules

Knowledge Representation in ES

- Rules
  - Rule-based systems
    - IF-THEN expressions
    - Some similarities to logical implication
    - More emphasis on what is done with rules than what they mean
      - i.e., procedural aspects emphasized rather than declarative ones

Rules: Basics

- Knowledge representation in the form of "rules"
  - IF-THEN expressions
    - If <condition(s)> then <conclusion>
  - Condition-action expressions
    - If <condition(s)> then <action>
- Effectively propositional or FOL implications
- Example
  - `If the traffic light is red AND you have stopped, THEN a right turn is OK`
- Biomedical example
  - `If patient has high levels of the enzyme ferritin in their blood AND patient has the Cys282-> mutation in HFE gene THEN conclude patient has hemochromatosis`
- Condition
  - aka premise, antecedent
- Conclusion
  - aka, action, result, consequent

Rules: Antecedent

- IF
  - Antecedent
    - A series of tests
      - Just like the tests of the nodes of a decision tree
  - Tests are usually logically AND’d together
    - Like the Horn clauses used for chaining inference
    - But may also be general logical expressions

Rules: Consequents

- THEN
  - Consequent
    - Conclusions, set of conclusions
    - Actions, set of actions
    - A probability distribution assigned by a rule specifying support/certainty in a given consequent
      - Reasoning with uncertainty
- Conflicts arise if different conclusions apply
  - Strategies needed to deal with conflict resolution

More on Consequents

- Relation
  - If the fuel tank is empty Then the car is dead
- Recommendation
  - If the season is autumn AND the sky is cloudy AND the forecast is drizzle Then the advice is "*take an umbrella*"
- Directive
  - If the car is dead AND the fuel tank is empty Then the action is "*refuel the car*"

More on Consequents

- Strategy
  - If the car is dead Then the action is "check the fuel tank"
  - Step1 is complete
    - If step1 is complete AND the 'fuel tank' is full Then the action is "*check the battery*"
    - Step2 is complete
- Heuristic (Prediction model - Classification/Regression)
  - If the spill is liquid AND the "spill PH" <6 AND the "spill smell" is vinegar Then the "spill material" is "acetic acid"

Summary: Characteristics of Rules

	First Part	Second Part
Names	Premise Antecedent Situation IF	Conclusion Consequence Action THEN
Nature	Conditions, similar to declarative knowledge	Resolutions, similar to procedural knowledge
Size	Can have many IFs	Usually only one conclusion
Statement	And Statements OR statements	All conditions must be true for a conclusion to be true If any condition is true, the conclusion is true

Simple rules with Relations

- We can also add a variety of conditional relations to rules (e.g., =, >, <, …)
- Rules comparing an attribute-value to a constant (e.g., temperature < 45)
  - These are called propositional rules
    - Same expressive power of propositional logic
  - Example
    - If credit is high AND salary is more than EUR30.000, OR assets are more than EUR7.000 AND pay history is not 'poor' Then approve the loan up to EUR10.000 and list the loan category 'B'
  - What if the problem involves relationships between attributes?
    - Can't be expressed with propositional rules
    - More expressive representation required

Relations between Attributes

- Standard relations =, <,>
- Comparing attributes with each other enables rules like this
  - If width > height Then lying
  - If height > width Then standing
- This description generalizes better to new data
- But searching for relations between attributes can be expensive
  - Simple solution: add an extra attribute
    - E.g., a binary attribute "is width < height?"
- Using variables and multiple relations (similar to FOL)
  - If height_and_width_of(x,h,w) AND h>w Then standing (x)

Rules as a Knowledge Base

- Two ways of executing a rule set
  - Ordered set of rules ("decision list")
    - Order is important for interpretation
  - Unordered set of rules
    - Rules may overlap and lead to different conclusions for the same instance
- Default rules
  - Common when using ordered rule sets
    - If is_top_of(x,z) AND height_and_width_of(z,h,w) AND h > w AND is_rest_of(x,y) AND standing(y) Then standing(x)
    - If empty(x) Then standing(x)

Rules as a model of Human Behavior

- Each rule in a knowledge base represents an autonomous chunk of expertise
  - Conceptually, individual rules are often logically OR'd together
- When combined and fed to the inference engine, the set of rules behaves synergistically
- Rules can be viewed as a simulation of the cognitive behavior of human experts
  - Rules represent a model of actual human behavior

Rule Inference

- Two principal tasks imply two control regimes
  - Forward chaining (data-driven)
    - Start with facts, determine applicable rules, and apply one
  - Backward chaining (goal-oriented)
    - Look for rules which decompose the goal
      - Solve smaller goals
- Is one better than the other?
  - No general answer is possible
    - Depends on the application

Rule Matching

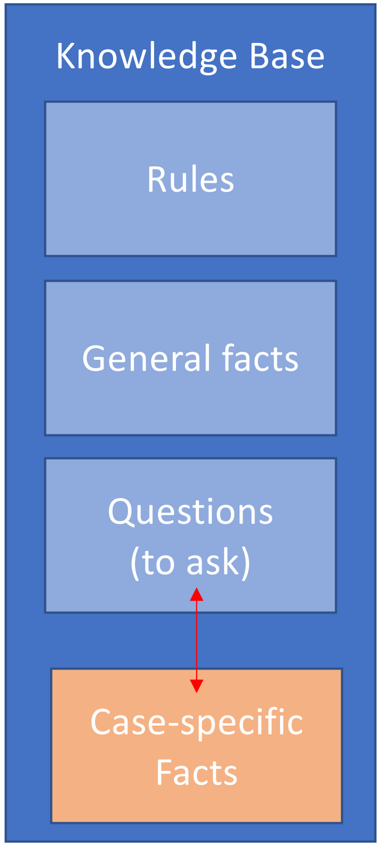
- Matching determines the applicable knowledge of the given situation
- Rules are *activated*, i.e., triggered, based on the type of inference/chaining used
  - Forward chaining
    - Match rule antecedents to fact base
  - Backward chaining
    - Match rule consequents to fact and rule base
- Matching of many rules against many facts is computationally expensive
- Matching often yields > 1 applicable rule
  - These rule instances are said to be in conflict
  - Use conflict resolution to pick one for firing (forward chaining)
- Firing
  - Executing the consequence of an applicable rule instance (forward chaining)
- Variables in rules get values during the matching process

Knowledge Base

- Holds the expertise that the system can deploy
  - Aka Production memory (in production systems)
- Uniquely developed for a specific expert task
- Exists independent from ES shell
- Includes long and short-term memory of the ES
  - Long-term
    - General facts/data and rules that don't change
  - Short-term (i.e., working memory)
    - Case-specific facts added to the KB by the inference engine in response to a query

Knowledge Base Elements

- Rule Base
  - Implication rules capturing domain knowledge
- Fact Base
  - General facts that do not change
  - Can reference a database of key information or variable values
- Question Base
  - Questions written in natural language can be specified here
    - Request case-specific facts from the user
    - The Inference engine utilizes question answers to determine if a given rule activates
- Case specific facts
  - Facts provided by the user for a given query
  - Can be given as question responses (when fewer facts are needed)
  - Can be loaded in bulk (when many unique facts are needed/available)



Rule Base

- A largely unordered set of non-repeating rules
- Depending on the task complexity
  - Rule bases for real-world problems may contain a few hundred to thousands of rules
- Using simple IF, AND, THEN rules
  - How to handle OR?
    - E.g., If A OR B OR C Then D
    - Break into individual rules
      - If A Then C
      - If B Then D
      - If C Then D
- Example rule
  - If son_of($son, $father) Then father_of($father, $son)

Fact Base

- An unordered set of facts that give values for propositions/predicate statements
- Facts should follow the syntax of rules so the inference engine can match facts to be applied to rules
- Example Rule
  - If son_of($son, $father) Then father_of($father, $son)
- Example Facts
  - son_of(Steve, Joe)
  - son_of(Bill, Dave)

Question Base

- An unordered set of questions applied by the ES when needed to give the system a more human, feel for user interaction
- Questions return the following user-entered values
  - Yes/No
  - True/False
  - Multiple choice (Pick one or pick all that apply)
  - Enter value(int, float, text)
- Example
  - who_father($ans)
  - Who is your father?
  - $ans= select_1
    - 1:Joe
    - 2: Steve
    - 3: Bill
    - 4: Dave
    - 5: I don't know

Expert Systems: Inference Engine

Rule-Based Expert Systems

- Rules are the most common representation in ES
  - In particular, simple IF-THEN expressions
    - Rule antecedents are only permitted to be conjunctions (AND's)
- The inference engine determines which rule antecedents are satisfied
  - The left-hand side must match a fact in the KB
  - Results in the right-hand side (consequent) being activated (fired)
  - An activated rule may generate new facts
  - The activation of one rule may subsequently lead to the activation of other rules

Inference Engine

- Also known as Rule interpreter
- The thinking part of the system
  - Controls how the IF-THEN rules are applied to the facts
  - Allows for the acquisition of further information from the user
    - Prompted for further input via natural language interface (perhaps via questions)
  - Can be used to
    - Come to some hypothesis/solution
    - Refine a hypothesis/solution
    - Resolve conflicts between currently competing hypotheses/solutions
- Links rules with facts to generate new facts
  - New facts represent conclusions about the state of the domain given the observations

How Expert Systems Work

- The inference engine implements a Forward Chaining-like algorithm, triggered by the addition of new facts in the working memory:

```
while there is some active rule do
      select one active rule (using conflict resolution strategies)
      execute the actions of the selected rule
```

- Three kinds of actions exist:
  - modifying one fact in the working memory
  - adding one fact to the working memory
  - removing one fact from the working memory

Recall: Rule Inference

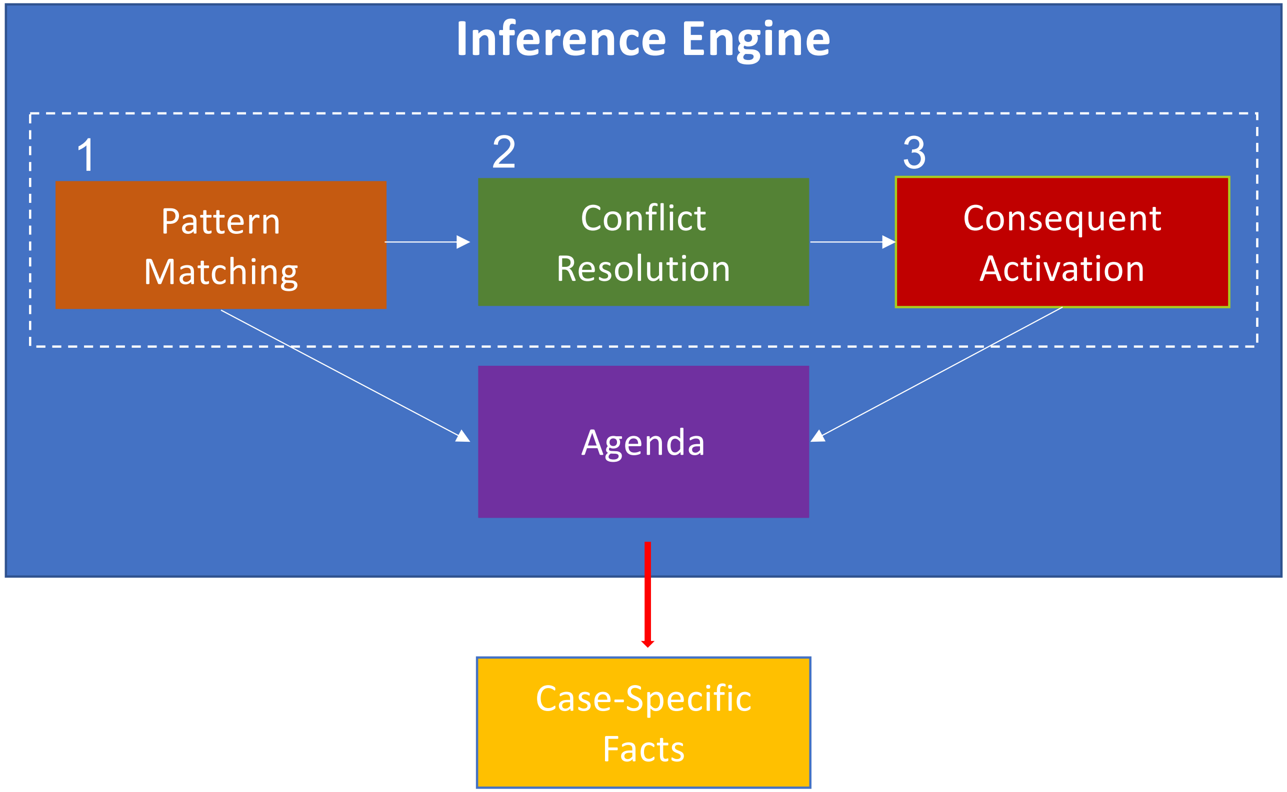
- Two principle tasks imply two control regimes
  - Forward chaining (data-driven)
    - Start with facts, determine applicable rules, and apply one
    - Best for prognosis, monitoring and control
  - Backward chaining (goal-oriented)
    - Look for rules which decompose goal
      - Solve smaller goals
    - Best for diagnosis problems
- Is one better than the other?
  - Depends on the application

More on Chaining

Forward Chaining	Backward Chaining
Data drive reasoning	Goal driven reasoning
Start with facts in the knowledge base	Start with a user-entered goal to be proved
Inference rules are applied by matching facts to the if part of rules in the knowledge base	Inference rules are applied by matching the goal to the then part of the rules in the knowledge base
Result in new facts added to the knowledge base	Result in subgoals to be proved
Rules run automatically when forward chaining is activated to assert new facts	Rules are directly used to prove the goal

Inference Engine Components

- Match-Resolve-Act Cycle aka Recognize-Act Cycle
  - Describes the execution of rules by the inference engine



Recall: Pattern/Rule Matching

1 Pattern Matching

- Matching determines the applicable knowledge in the given situation
  - This is the search component of the system
- Rules are 'activated', i.e., triggered, based on the type of inference/chaining used
  - Forward chaining
    - Match rule antecedents to fact base
  - Backward chaining
    - Match rule consequents to fact and rule base
- Matching many rules against many facts can be computationally expensive
- Matched rules are added to the agenda
  - A prioritized list of rules created by the inference engine

Variable Binding

- Binding
  - Variables in rules acquire values during the matching process
- Antecedents and consequents may contain variables
  - We assume variables look like ($x)
- Example
  - Rule: If son_of($son, $father) Then father_of($father, $son)
  - Facts: son_of(Steve,Joe), daughter_of(Mary,Joe)
- Let's assume forward chaining
  - Example rule matches son_of(Steve,Joe)
  - Thus 'Steve' is bound to $son and 'Joe' is bound to $father
  - When the consequent of this rule is activated
    - father_of(Joe, Steve) added to case-specific facts (i.e., working memory)

Variable Binding

- Once a variable is bound, that variable is replaced by its binding wherever it appears in the same or subsequently processed patterns
- Whenever the variables in a pattern are replaced by their bindings, the pattern is said to be instantiated
  - Recall this from logic?
- In forward chaining, bindings typically persist as new facts
- In backward chaining, bindings can be temporary as hypotheses that are being tested

Conflict Resolution

2 Conflict Resolution

- Matching often yields > 1 applicable rule (put in agenda)
  - Which to run first, or do we only want one?
    - E.g., we may only want medication to be prescribed
  - Assuming we only want one 'action' to fire
    - These rule instances are said to be in conflict
    - Use conflict resolution to pick one for firing (forward chaining)
- Select the rule with the highest priority from the agenda
- Different strategies may be used to resolve such conflicts

Conflict Resolution Strategies

- Order (i.e., source file ordering)
  - Pick the first rule that matches in the rule base
    - Not ideal for very large rule-sets
- Specificity
  - Pick the most specific rule
    - If the conditions of one rule are a subset of a second rule, choose the second rule
      - E.g., (IF A AND B THEN C) vs (IF A THEN C) -> pick the first!
- Lexical order
  - E.g., alphabetical/dictionary order
    - Somewhat arbitrary, but may be useful in some domains

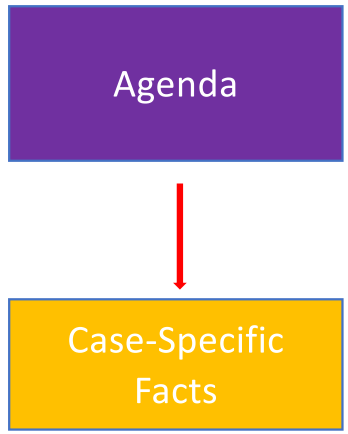
Conflict Resolution Strategies

- Importance
  - Pick rule with highest priority
  - Rules are defined with a priority score
  - Generally discouraged as this can be very hard to set up precisely in large rule bases
  - Can be controlled by meta-rules
  - Example
    - IF patient has pain, THEN prescribe painkillers (priority 10)
    - IF patient has chest pain, THEN treat for heart disease (priority 100)
- Recency
  - Pick the rule most recently added or modified
  - Or that has antecedents most recently added or modified
  - Require time-tags
- Refactoriness
  - Don't allow the same binding instance of a rule to fire again

Consequent Activation

3 Consequent Activation

- Once a rule on the agenda is selected …
- Rule activation/Firing/triggering
  - Executing the consequence of an applicable rule instance (forward chaining)
    - Resulting new facts added to case-specific facts
  - Remove the rule from the agenda
- The cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

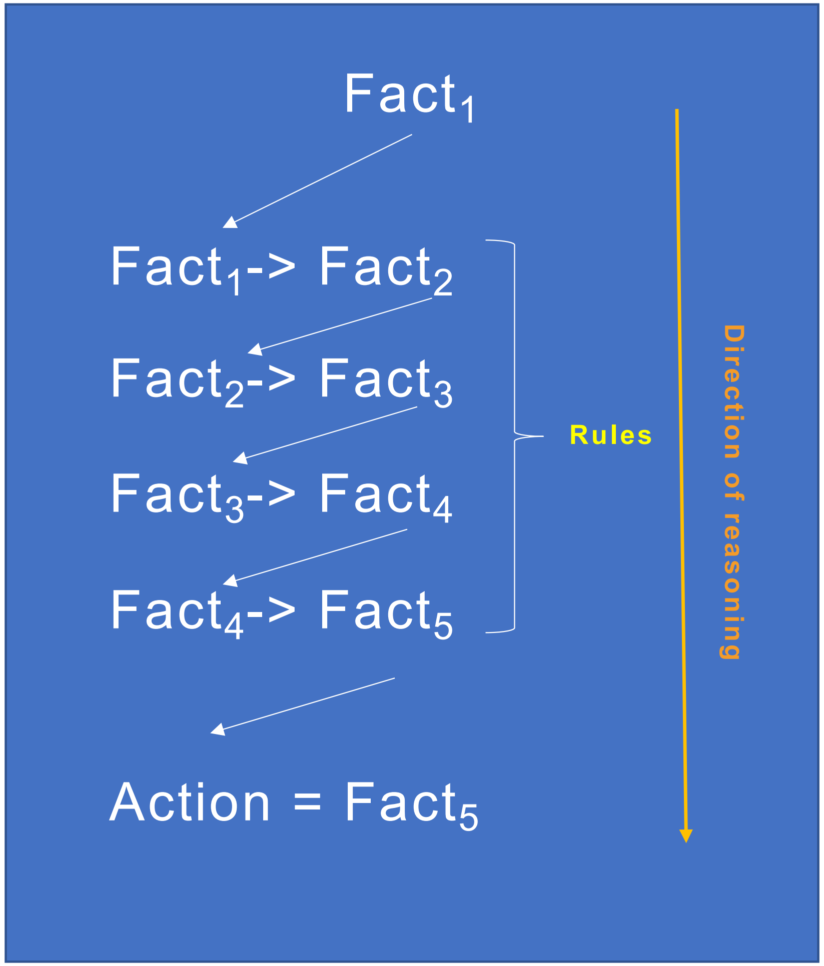


Forward Chaining with Rules

Repeat

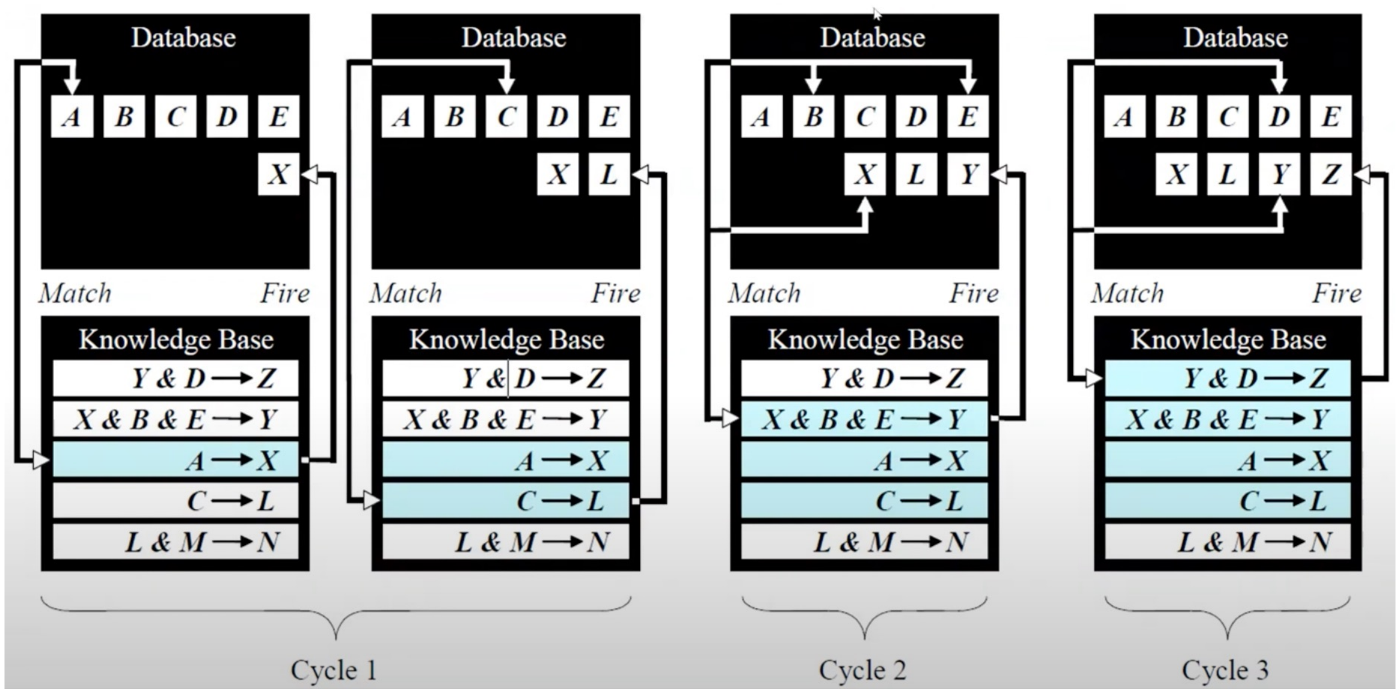
- Apply all the rules to the current facts
- Each rule firing may add new facts

Until no new facts are added



Forward Chaining Walkthrough

- Goal is to prove Z



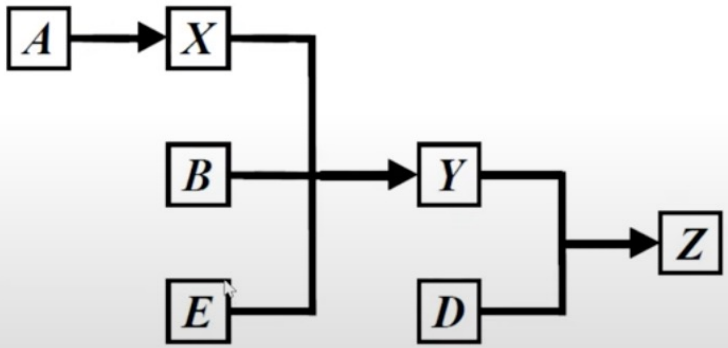
Chaining

- Forward chaining yields an inference chain
    - A, B, and E are facts

Rule 1: IF Y is true
AND D is true
THEN Z is true

Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true

Rule 3: IF A is true
THEN X is true



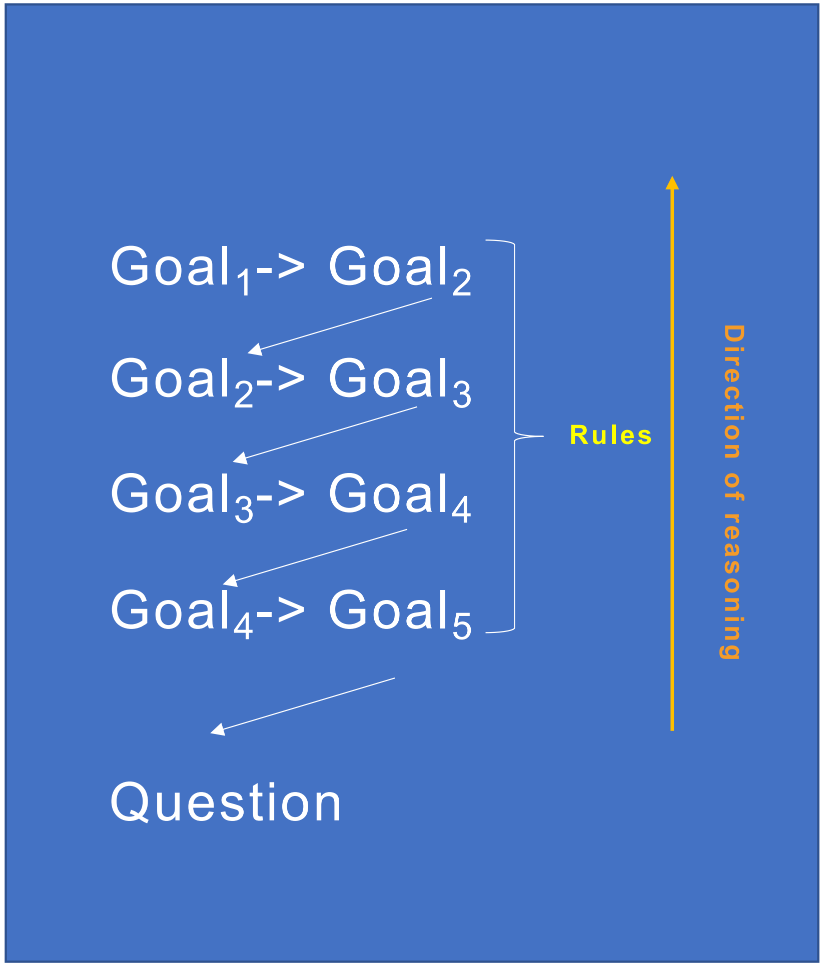
Backward Chaining with Rules

Repeat

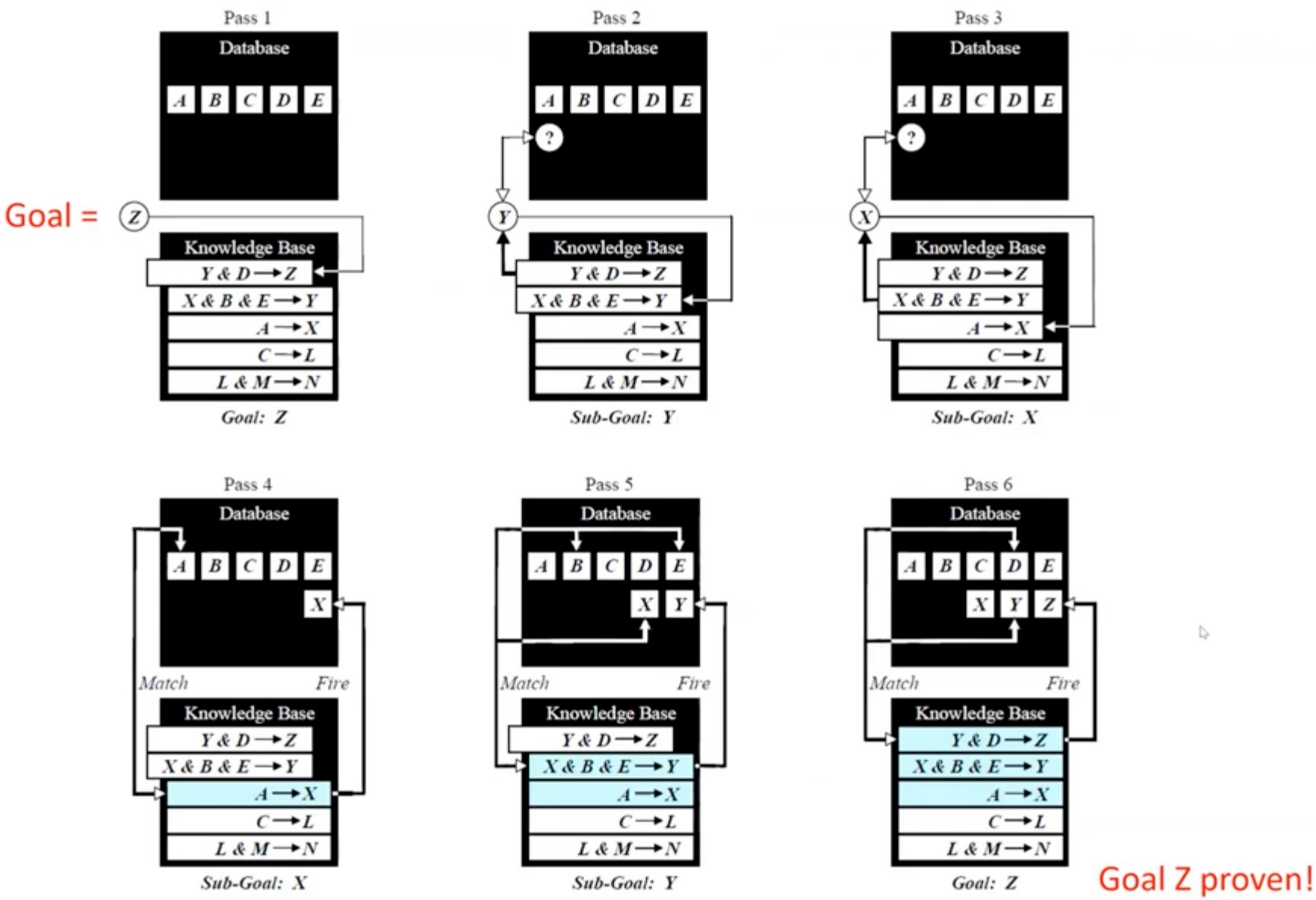
- See if the consequent is in fact base
- If not, find rules with the target
- Such rules stack
    - Its antecedents become new target consequents (i.e., subgoal)

Until a connected set of subgoals are satisfied by facts (success)

- Or no connected set of subgoals can be satisfied by the facts



Backward Chaining Walkthrough



Failure of Query

- The interpretation of failure as negation
  - Failure to prove the query suggests it's not true
  - However, could also mean the database was incomplete or inadequately maintained
- Under some circumstances, there is a possibility of infinite looping when attempting to retrieve missing information
- In these cases
  - Necessary to apply termination criteria that explicitly detect this and allows the system to fail gracefully

Mixed Chaining

- Some ES's can apply forward and backward chaining simultaneously
- The strategy is for the system to chain in one direction, then switch to the other direction, so that
  - The diagnosis is found with maximum efficiency
  - The system's behavior is perceived as human

Inference Engine Efficiency

- How do we reduce the cost of matching?
- Markov algorithms
  - Rules with higher priorities are applied first
  - Allows more efficient execution of production systems
  - But still not efficient enough for expert systems with large sets of rules
- Save intermediate match information (RETE)
  - Share intermediate match information between rules
  - Re-compute intermediate information for changes
  - Require extra memory for intermediate match information
  - Scales well to large rule sets
- Re-compute match for rules affected by change (TREAT)
  - Check changes against rules in conflict set
  - Less memory than RETE
  - Doesn't scale as well to large rule sets

Expert System: Other Components

User Interface

- The means of interaction between a user seeking a solution and an ES
  - Edit facts/rules
  - Query the system
  - Answer questions
  - Be presented with solution(s) and explanations
- UI is directly or indirectly connected to all parts of the ES
- UI may be
  - Simple Question/Answers
  - Menu drive, e.g., GUI
  - Accommodate NLP
    - So it can easily be used by a person well-acquainted with the application area but not necessarily AI systems

Knowledge Base Editor

- Also known as Knowledge acquisition facility
- Automatic way for the user to enter knowledge in the system bypassing the need for explicit coding expertise
- Facilitates
  - Integration of new knowledge
  - Editing of existing knowledge

Explanation System (Justifier)

- Explains the reasoning of ES to the user
  - How a particular conclusion was reached (reasoning chain)
    - Which rules have fired, which facts deduced
  - Why was another conclusion not reached?
  - Why a specific fact is needed?
  - Why a fact wasn't used?
- For instance, a physician has to understand the reasoning for a recommendation
- A major advantage for ES over NN-based systems

Expert Systems Shells

- Software platform for building expert systems
  - Developers only need to generate the knowledge base
- Simplifies the process of creating an expert system in a faster, cheaper, and more efficient way
- Must use a representation that the shell's inference engine can reason with
  - Rules
- Different shells may include different components, terminology, coding language and different syntax for implementing rules/facts, etc
- Examples
  - CLIPS, PyCLIPS, PyKE, Knowledge Pro, ..

Expert Systems Shells: CLIPS

- C Language Integrated Production System (CLIPS)
  https://www.clipsrules.net/
- Developed since 1984 by the Johnson Space Center, NASA
- Now maintained independently from NASA as public domain, open source, free software
- Currently used by government, industry, and academia
- Main features:
  - interpreted, functional language (object-oriented extension)
  - specific data structures and instructions/functions for expert system implementation
  - interfaces to other languages (C, Python, etc.)