Motivation: Expert Systems

- **MYCIN**: medical diagnosis (bacterial infection)
  
  if has(Patient, Infection5) and 
  no_apparent(Organism7, BloodTest) 
  then conclude(Staphylococcus) with certainty 0.8

- **XCON**: computer configuration
  
  if selected(PrinterSt800, ComputerDPK202)
  then add(PrinterConversionCard4)

AI in Practice: Expert System

Expert Systems in the Computer Industry

Digital Equipment Corporation had a complicated problem in their shipping department. Each customer order was generally for a variety of computer products, but was sometimes accompanied by requests for special conversion cards to make two or more products compatible with another. Sometimes different products or special connection cards were required. Although there were rules about what computers could be attached to each other, they were difficult to remember, and the customer's order had to be tested for compatibility before being shipped.

A computer program was written to solve this problem; an expert system called XCON. XCON was designed to match computers and their peripherals, it checks prerequisites such as cabling and support software based on rules. Other expert systems design local area networks, assess the computing needs of the customer, and plan the physical placement of the computers.

Because Digital's computer sales and products are always changing, the rules for XCON need to change frequently. To facilitate this, rules in XCON are simple. One rule might be "If the Blackwatt 600 printer and DPK202 computer have been selected, add a printer-conversion card, because they are not compatible." Together, these rules form an expert system that has been used successfully for more than a decade.
3.1 Propositional Logic

- Logic: a mathematical tool for constructing and manipulating symbolic expressions.
- Propositional logic:
  - symbols representing propositions ("the light is on" or "the door is closed")
  - symbols representing ways of combining propositions ("the light is on and the door is closed")
- Logics are characterized in terms of languages.
- Logical languages are declarative ("what is") rather than imperative or procedural ("how to").

Syntax for P (1/2)

- A formal language is defined by a set of well-formed formulas (wffs):
- A set of propositional variables
- The connectives \( \lor, \land, \neg, \rightarrow \), and \( \leftrightarrow \)
- A set of wffs inductively defined as
  - The propositional variables
  - \( A_1 \lor A_2 \lor ... \lor A_n \)
  - \( A_1 \land A_2 \land ... \land A_n \)
  - \( \neg A \)
  - \( A \rightarrow B \) and \( A \leftrightarrow B \)

Syntax for P (2/2)

- Conjunction and Conjunct:
  - \( A_1 \land A_2 \land ... \land A_n \)
- Disjunction and Disjunct:
  - \( A_1 \lor A_2 \lor ... \lor A_n \)
- Literals:
  - \( P \) and \( \neg P \) are literals associated with the proposition \( P \).
  - positive literals \( (P) \) and negative literals \( (\neg P) \).
- Rule:
  - A rule is of the form \( A \rightarrow B \), where \( A \) and \( B \) are either positive literals or conjunctions of positive literals.

Semantics for P (1/2)

- Interpretation:
  - a possible association of the symbols in a language with the elements of the domain.
  - an assignment to each propositional variable either True or False
- Model:
  - An interpretation \( I \) is a model for a set of formulas if \( I \) assigns True to each formula in the set.
- Satisfiable:
  - A formula is satisfiable if it has a model.
- Valid:
  - A formula is valid if its negation is not satisfiable.
Theory:
- A set of formulas

Valid w.r.t. a theory:
- A formula \( A \) is valid w.r.t. a theory \( T \) if \( A \) is True in all the models of \( T \).

Inconsistent:
- A theory (or formula) is model theoretically inconsistent if it has no models.

Tautology:
- A formula that is always true is called a tautology.

Contradiction:
- A formula that is always false is called a contradiction.

Normal Forms
- CNF (Conjunctive Normal Form)
  - Conjunction of disjunctions of literals
  - e.g.: \( P \rightarrow Q, \neg(S \land \neg T) \), and \( R \)
    \( \iff (\neg P \lor Q) \land (S \lor T) \land R \)
- DNF (Disjunctive Normal Form)
  - Disjunction of conjunctions of literals

Rulebase:
- A set of rules of the form \((P_1 \land \ldots \land P_n) \rightarrow Q\)
  - \(P_i\): antecedents, \(Q\): consequent
- Horn clause: a disjunction with one positive literal
  \((P_1 \land \ldots \land P_n) \rightarrow Q \iff (\neg P_1 \lor \ldots \lor \neg P_n \lor Q)\)

Rules of Inference
- Modus Ponens:
  - For any wffs \( A \) and \( B \), given \( A \) and \( A \rightarrow B \), we can conclude \( B \).
- Conjunction:
  - Given wffs \( A_1 \) through \( A_n \), we are warranted in concluding \((A_1 \land \ldots \land A_n)\)
- Resolution:
  - Given \( A_1 \land \ldots \land A_i \land \neg C \land A_{i+1} \land \ldots \land A_m \)
    and \( B_1 \land \ldots \land B_j \land C \land B_{j+1} \land \ldots \land B_n \)
    Conclude \( A_1 \land \ldots \land A_i \land B_1 \land \ldots \land B_n \)
Proofs and Theorems

- **Proof**: A proof is a sequence of statements in an appropriate language, where each statement is an axiom or an immediate consequence of some rule of inference and some prior statements in the sequence.
- **Theorem**: Each statement in a proof is a theorem of the formal system.

A Simple Proof

- **Given**: axioms, \{ \((P \land Q) \Rightarrow R\), \((S \land T) \Rightarrow Q\), S, T, P \}, two rules of inference (conjunction and modus ponens)
- **Proof of R**:
  1. S  **AXIOM**
  2. T  **AXIOM**
  3. \(S \land T\)  **CONJ: 1, 2**
  4. \((S \land T) \Rightarrow Q\)  **AXIOM**
  5. Q  **MP: 3, 4**
  6. P  **AXIOM**
  7. \(P \land Q\)  **CONJ: 5, 6**
  8. \((P \land Q) \Rightarrow R\)  **AXIOM**
  9. R  **MP: 7, 8**

Completeness, Soundness, and Decidability

- **Completeness**: A formal system \(S\) is **complete** if all the formulas valid with respect to the axioms of \(S\) are also theorems of \(S\).
- **Soundness**: A formal system is **sound** if all its theorems are valid.
- **Decidability**: A procedure is **effective** if it obtains a correct answer in a finite number of steps.
  A formal system is **decidable** if there exists an effective procedure for answering the question for any formula whether or not that formula is a theorem.

Solving Problems with Logic: The Housing Lottery Problem

- **Given** the following constraints, how the following four students are ranked?
  1. Lisa is not next to Bob in the ranking.
  2. Jim is ranked immediately ahead of a biology major.
  3. Bob is ranked immediately ahead of Jim.
  4. One of the women is a biology major.
  5. Mary and Lisa is ranked first.
Translation of English into Logic (1/2)

1. “Lisa is not next Bob in the ranking”:
   \[(\neg \text{Lisa} \text{ ahead of Bob}) \land 
   \neg \text{Bob} \text{ ahead of Lisa}\]

2. “Jim is ranked immediately ahead of a biology major”:
   \[\text{(Jim} \text{ ahead of Mary} \lor \text{Mary} \text{ bio major}) \lor 
   \text{(Jim} \text{ ahead of Bob} \lor \text{Bob} \text{ bio major}) \lor 
   \text{(Jim} \text{ ahead of Lisa} \lor \text{Lisa} \text{ bio major})\]

3. “One of the women is a biology major”:
   \[\text{Mary bio major} \lor \text{Lisa bio major}\]

Translation of English into Logic (2/2)

4. “Bob is immediately ahead of Jim”:
   \[\text{Bob} \text{ ahead of Jim}\]

5. “Mary or Lisa is ranked first” (\(\text{MB} = \text{M ahead of Bob}\) etc.):
   \[\text{(MB} \lor \text{BL} \lor \text{LJ}) \lor \text{(MB} \lor \text{BJ} \lor \text{JL}) \lor \text{(ML} \lor \text{LB} \lor \text{BJ}) \lor \text{(ML} \lor \text{LJ} \lor \text{JB}) \lor \text{(MJ} \lor \text{JL} \lor \text{LB}) \lor \text{(MJ} \lor \text{JB} \lor \text{BL}) \lor \text{(LB} \lor \text{BJ} \lor \text{JM}) \lor \text{(LB} \lor \text{BM} \lor \text{MJ}) \lor \text{(LM} \lor \text{MJ} \lor \text{JB}) \lor \text{(LM} \lor \text{MB} \lor \text{BJ}) \lor \text{(LB} \lor \text{JL} \lor \text{ BM}) \lor \text{(LB} \lor \text{JM} \lor \text{ MB})\]

Representing Common-Senses

CS1: “A student can be immediately ahead of at most one other student”
   \[\text{Bob} \text{ ahead of Jim} \rightarrow \neg \text{Bob} \text{ ahead of Lisa}\]

CS2: “If X is immediately ahead of Y then Y cannot be immediately ahead of X”
   \[\text{Bob} \text{ ahead of Jim} \rightarrow \neg \text{Jim} \text{ ahead of Bob}\]

Reasoning

- Given: \(\text{Bob} \text{ ahead of Jim}\) : known fact (3)
  and:  \(\neg \text{Bob} \text{ ahead of Lisa}\)
  Conclude:  \(\neg \text{Bob} \text{ ahead of Lisa}\) : Modus Ponens
- It follows that
  \[\neg (\text{Mary} \text{ ahead of Bob} \lor \text{Bob} \text{ ahead of Lisa} \lor 
  \text{Lisa} \text{ ahead of Jim})\]
- Likewise, eliminate all the disjuncts in which Bob is not ahead of Jim!
3.3 Automated Theorem Proving in P

- **Axioms:**
  - rules of the form \((P_1 \land ... \land P_n) \implies Q\).
  - \(P_i\) and \(Q\) are propositional variables.

- **Fact:**
  - rules without antecedents
  - just a propositional variable

- **Goals:**
  - conjuncts of the conjunction

---

Goal Reduction in P

- Given a goal \(Q\) and a rule \((P_1 \land ... \land P_n) \implies Q\),
  if we can prove \((P_1 \land ... \land P_n)\), then we can prove \(Q\).
- Proving \(Q\) is *reduced* to proving \((P_1 \land ... \land P_n)\).
- **Homework:**
  - implement the simple propositional theorem prover theorem in p. 86 and show the proof procedure for the example shown in p. 87.

---

Theorem Proving: An Example

- \((P \land Q \implies R)\) can be represented as \((R \iff P \land Q)\) in Lisp

  > (setq rules `(if R (P Q) (R S T)) (P Q) (R S T))
  > (theorem `R rules)
  > T

- \(R \iff P \land Q\)

  \(Q \iff S \land T\)

---

3.4 Predicate Calculus

- Predicate logic (PC) is more expressive than propositional logic.
- PC allows for references to arbitrary objects, properties of objects, and relations among objects:
  - block17, table45, red(block17), on(block17, table45)
- PC allows for making quantified statements about arbitrary classes of objects:
  - \(\forall x, \text{on}(x, \text{table45}) \implies \text{red}(x)\)
Syntax for PC (1/2)

- **Basic Entities:**
  - A set of *predicates* of arity \( n \)
  - A set of *functions* of arity \( n \)
  - A set of *constants* and a set of *variables*

- **Terms** are constructed from functions, constants, and variables:
  - Constants and variables are terms.
  - \( f(t_1, t_2, \ldots, t_n) \) is a term, where \( f \) is a function and \( t_i \) are terms.

Syntax for PC (2/2)

- **Wffs** are constructed from terms, predicates, and quantifiers:
  - Atomic sentence \( p(t_1, t_2, \ldots, t_n) \), where \( p \) is a predicate and \( t_i \) are terms.
  - \( A_1 \land A_2 \land \ldots \land A_n \), where \( A_i \) is a wff.
  - \( A \lor A_2 \lor \ldots \lor A_n \)
  - \( \neg A \)
  - \( A \implies B \) and \( A \iff B \)
  - \( \forall x_1, x_2, \ldots, x_n \) and \( \exists x_1, x_2, \ldots, x_n \)

Translating English into Logic

- "All purple mushrooms are poisonous."
  \( \forall x. (\text{mushroom}(x) \land \text{purple}(x)) \implies \text{poisonous}(x) \)

- "No purple mushroom is poisonous."
  \( \forall x. (\text{mushroom}(x) \land \text{purple}(x)) \implies \neg \text{poisonous}(x) \)

- "All mushrooms are either purple or poisonous."
  \( \forall x. (\text{mushroom}(x) \implies ((\text{purple}(x) \lor \text{poisonous}(x))) \)

3.5 Formal System for PC

- **Rules of Inference:**
  - Modus ponens
  - Conjunction
  - Universal instantiation

- **Universal instantiation**
  - substitute any term for all occurrences of a universally quantified variable.
Proof in PC: An Example

- **Given**: person(fred), location(fred, lobby).
  
  \[ Vx,y, (person(x) \land location(x,y)) \implies occupied(y) \]

- **Proof**: occupied(lobby)
  
  1. person(fred) \hspace{1cm} Axiom
  2. location(fred, lobby) \hspace{1cm} Axiom
  3. \[ Vx,y, (person(x) \land location(x,y)) \implies occupied(y) \] \hspace{1cm} Axiom
  4. \[ Vx, (person(fred) \land location(fred, y)) \implies occupied(y) \] \hspace{1cm} UI: x \rightarrow fred, 3
  5. \[ (person(fred) \land location(fred, lobby)) \implies occupied(lobby) \] \hspace{1cm} UI: y \rightarrow lobby, 4
  6. \[ (person(fred) \land location(fred, lobby)) \] \hspace{1cm} CONJ: 1, 2
  7. occupied(lobby) \hspace{1cm} MP: 5, 6

Eliminating Quantifiers: Skolemization

- **Example 1**:
  
  \[ E(x, happy(x)) \implies E(x, happy(\text{the-happy-one})) \]

- **Example 2**:  
  
  \[ Vx, Ey, loves(x,y) \implies Vx, loves(x,\text{the-love-of}(x)) \]

- **Skolemization**:  
  
  \[ Vx_p, ..., x_n, Ey, \phi(x_p, ..., x_n, y) \implies Vx_p, ..., x_n, \phi(x_p, ..., x_n, \text{skm}(x_p, ..., x_n)) \]

  where \text{skm} is a new function (Skolem function).

- **Example 3**:  
  
  \[ Vx, Ey, location(y,x) \implies Vx, location(\text{sk37}(x),x) \]

Learning and Deductive Inference

- Learning (inductive inference): draws general conclusions from particular examples
- Deductive inference: conclusions necessarily follow from axioms according to specified rules of inference.

A Softbot for UNIX

- The softbot knows that a superuser can delete any file and that it is a superuser.

  \[ Vf, u, super(u) \implies delete(f, u) \]

  super(bot9)

- Using these axioms, the softbot can prove that it can delete any particular file.

  1. super(bot9) \hspace{1cm} Axiom
  2. \[ Vf, u, super(u) \implies delete(f, u) \] \hspace{1cm} Axiom
  3. \[ Vf, super(bot9) \implies delete(f, bot9) \] \hspace{1cm} UI: u \rightarrow bot9, 2
  4. super(bot9) \implies delete(tetris, bot9) \hspace{1cm} UI: f \rightarrow tetris, 3
  5. delete(tetris, bot9) \hspace{1cm} MP: 1, 4
Decidability of PC

- The predicate calculus is not, in general, decidable. That means that there is no effective method for deciding whether a given formula is a theorem.
- However, there are specific theories that are decidable.
- For example, any theory with a finite number of terms is decidable.

3.6 Automated Theorem Proving in PC

- Matching and Universal Instantiation
- AND/OR Trees
- Proof Trees (AND/OR Proof Trees)
- Unification
- Semantic Networks

Matching

- *Universal instantiation* allows us to substitute any term for all occurrences of a universally quantified variable in a formula.
- We say that two formulas *match* if we can find substitutions for the variables appearing in the formula such that the two are syntactically equivalent.
- Example:
  - `(pleasing block19)`
  - `(pleasing (? x)) ;; x universally quantified`

Algorithm for Matching Constant and Pattern Expression

1. If the *set of bindings is nil*, return nil (failure).
2. If *P* is an *atom*, then, if *P* and *C* are eq, return the list of bindings (success); otherwise, return nil (failure).
3. If *P* is a *variable*, then, if it is already bound, match *C* against the binding for *P*; otherwise return a new list of bindings.
4. (*P* is non-nil list structure) If *C* is *nil*, return nil.
5. (Both *P* and *C* are non-nil list structure) Match the first of *P* and the first of *C* using the bindings that we obtain from matching the rest of *P* and the rest of *C* using the bindings that we started with.
Goal Reduction in PC and AND/OR Trees

- An AND/OR tree is divided into alternating AND and OR layers.
- The edges through an AND layer connect a conjunction to its conjuncts.
- The edges through an OR layer connect a conjunct to possible reductions corresponding to rules in the database.

Theorem Proving: A Simple Example

- Consider the following theory:
  red(block17)
  sphere(block17)
  red(block41)
  cube(block41)
  \( \forall x, (\text{red}(x) \land \text{smooth}(x)) \Rightarrow \text{pleasing}(x) \)
  \( \forall x, \text{heavy}(x) \Rightarrow \text{pleasing}(x) \)
  \( \forall x, \text{sphere}(x) \Rightarrow \text{smooth}(x) \)

- Prove:
  pleasing(block17)

Figure 3.11: AND/OR Trees
Unification

- Unification is a more general form of matching.
- In unifying two patterns, we attempt to find the most general substitution that renders the two patterns syntactically equivalent.
- Example:
  \[ s = \emptyset \]
  \[ p: \text{loves}(\text{dog}(z), \text{dog}(\text{fred})) \]
  \[ q: \text{loves}(x, x) \]
  \[ \implies s = \{ x/\text{dog}(z) \} \]

Unification Procedure

- Procedure \text{unification}(p, q, s)
  1. Find the first disagreement between \( p \) and \( q \).
  2. If there is no disagreement, signal success and return \( s \).
  3. If there is a disagreement, then see if it can be resolved by making appropriate substitutions for variables.
     3.1 If neither terms involved is a variable, then fail.
     3.2 If at least one term is a variable, create \( p' \) and \( q' \) by substitutions.
     3.3 Create a new set of bindings, \( s' \), by adding the new binding to \( s \).
  4. Call \text{unification}(p', q', s') recursively.

Unification: Examples

- Examples (success):
  \[ s = \emptyset \]
  \[ p: \text{loves}(\text{mother}(z), z) \]
  \[ q: \text{loves}(x, y) \]
  \[ \implies s = \{ x/\text{mother}(z), y/z \} \]

- Examples (failure):
  \[ s = \emptyset \]
  \[ p: \text{loves}(z, z) \]
  \[ q: \text{loves}(\text{mother}(x), \text{fred}) \]
  \[ \implies s = \{ z/\text{mother}(x), z/\text{fred} \} \]

Goal Reduction with Quantified Formulas

- Consider the following theory:
  \[ \text{red}(\text{block}17) \]
  \[ \text{sphere}(\text{block}17) \]
  \[ \text{red}(\text{block}41) \]
  \[ \text{cube}(\text{block}41) \]
  \[ \forall x, (\text{red}(x) \land \text{smooth}(x)) \Rightarrow \text{pleasing}(x) \]
  \[ \forall x, \text{sphere}(x) \Rightarrow \text{smooth}(x) \]
- Prove:
  \[ \text{pleasing}(x) ; \text{quantified variable} \]
A class is a set of objects.

An instance of a class is a member of the corresponding set.

One class is a subclass of another class (called the superclass) if the set corresponding to the first is a subset of the set corresponding the second.

A concept description language is a specialized language that allows us to represent classes of objects, instances of classes, subclass relationships involving classes, and properties of classes and instances.

Consider the following set of formulas:

- instance(Fred, human)
- instance(Lisa, human)
- instance(Ralph, robot)
- subclass(robot, autonomous-system)
- subclass(human, autonomous-system)
- feature(human, construction, biological)
- feature(robot, construction, mechanical)
- feature(autonomous-system, behavior, adaptive)
3.7 Nonmonotonic Logic

- A logic is **monotonic**, if a formula is a theorem for a particular formal theory, then that formula is still a theorem for any augmented theory obtained by adding axioms to the theory.

- In a **nonmonotonic logic**, if a formula is a theorem for a formal theory, then that formula need not be a theorem for any augmented theory.

- Much of commonsense reasoning is nonmonotonic.

Closed-World Assumption (CWA)

- A closed-world assumption is often used to justify drawing a conclusion based on a lack of information:
  1. You want to decide if $\text{employee(Fred)}$.
  2. There is no record of Fred in your database.
  3. This does not mean you can prove $\text{employee(Fred)}$.

- Two special cases of CWA:
  - **Domain-closure assumption**: the only objects in the domain are the constants and functions specified in the theory.
  - **Unique-names assumption**: assumed unequal unless proved equal.

Abductive and Default Reasoning

- Nonmonotonic conclusions can be drawn by adding special-purpose rules of inference.

- Two special forms of nonmonotonic reasoning:
  - Default rule of inference (default reasoning)
  - Abductive rule of inference (abductive reasoning)

- Definition: For any wff $p$, let $C(p)$ follow if adding $p$ as an axiom does not result in a contradiction.

- The symbol $C$, standing for consistent, is a pseudo-predicate (cannot be proven)
**Default Rule of Inference**

Let $p$ and $q$ wffs.
1. $p$
2. $C(q)$
3. $q$

---

**Abductive Rule of Inference**

- Abduction warrants concluding the antecedent of an implication given the consequent if doing so is consistent.

1. $p \Rightarrow q$
2. $q$
3. $C(p)$
4. $p$

---

**3.8 Deductive Retrieval Systems**

- Deductive retrieval system: any system that stores knowledge in rules and draws conclusions from that knowledge.
- Most expert systems can be characterized as deductive retrieval systems.
- System architecture
  - Knowledgebase (KB): a database of facts and rules
  - Inference Engine (IE): a collection of procedures that operate on the database

---

**Knowledgebase**

- The database is more than a passive repository for facts and rules.
- In some KB, predicate calculus formulas are stored in a structure called a discrimination tree.
- The knowledgebase does not simply grow monotonically; information can be deleted.
- This requires that we have to retract some conclusions when some data they depend on are removed (necessity of reason maintenance).
Inference Engine

- Inference in deductive retrieval systems is initiated by adding and deleting information and by asking queries.
- Forward and backward chaining: standard methods for inference in deductive retrieval systems.
- Forward chaining: if \(<condition> \rightarrow <action>\)
  \-------------->
- Backward chaining: if \(<antecedents> \rightarrow <consequent>\)
  \<----------------
- Both methods can be used in general. However, for solving a specific problem one is more appropriate than the other.

Forward Chaining

- if \((P_1 \land P_2 \ldots \land P_n) \rightarrow Q\) then A
  “if you have added \(P_1\) through \(P_n\) to the database, then you should add \(Q\) to the db as well.”
- XCON: An expert system for computer configuration
  if selected(PrinterSt800, ComputerDPK202) then add(PrinterConversionCard4)

Backward Chaining

- if \((P_1 \land P_2 \ldots \land P_n) \rightarrow Q\) then \(Q\)
  “If you want to prove \(Q\), it suffices to prove \(P_1 \land P_2 \ldots \land P_n\)”
- MYCIN: An expert system for medical diagnosis
  if has(Patient, Infection5) and not_apparent(Organism7, BloodTest) then conclude(Staphylococcus) with certainty 0.8

Reason Maintenance Systems

- Why reason maintenance?
  - \(Q\) is added to the DB on the basis of \(P\) and \(P \rightarrow Q\) (i.e., modus ponens).
  - After \(P\) is removed (by some rule), \(Q\) should be removed (since it’s not true any more).
  - Automatically?
- How to maintain the truth of the system?
  - Dependency graph: specifies how items that might be stored in a DB depend on one another (Fig. 3.17).
Figure 3.17: Premises and Justifications

Exercises

- Homework
  - Problems 3.1-3.5
  - Problems 3.7-3.11