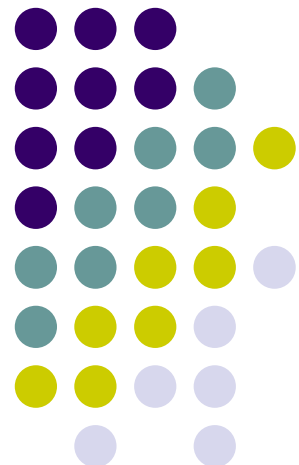
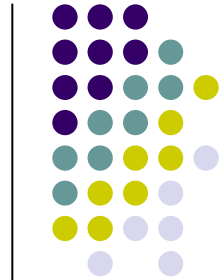
# Knowledge Representation and Conceptual Modelling IFT 63676

## MODULE 2: Logics and Knowledge Representation



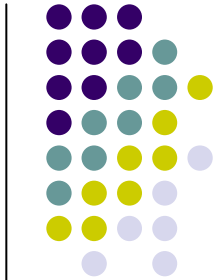
Bernard Moulin, Full Professor
Laval University
Computer Science and Software Engineering Department
September 2006

# Module Outline



- Introduction to Propositional Logic
- Propositional logic: Semantics
- Truth Tables for Connectives
- Limits of Propositional Logic
- Predicate Logic (Introduction)
- Predicate Logic (Syntax)
- Predicate Logic Semantics
- Proof and Inference
- Creation of a FOL Knowledge Base
- Introduction to Reasoning with Expert Systems
- Fundamental Characteristics of Expert Systems
- Forward and Backward Chaining
- About Data Bases and Expert Systems

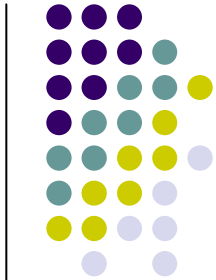
# Introduction to Propositional Logic



- We need formal a notation to represent knowledge,
  allowing automated inference and problem solving
- One popular choice is to use  logic
- Propositional logic is the simplest form of logic
  - Symbols represent facts: P, Q, etc.
  - These are joined by logical connectives (and, or, implication)

    $P \wedge Q$; $P \vee Q$ ; $Q \rightarrow R$
  - Given some statements in the logic we can deduce new facts
  - The proposition symbols P1, P2, etc. are also called sentences
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S1 and S2 are sentences, S1 $\wedge$ S2 is a sentence (conjunction)
  - If S1 and S2 are sentences, S1 $\vee$ S2 is a sentence (disjunction)
  - If S1 and S2 are sentences, S1 $\rightarrow$ S2 is a sentence (implication)
  - If S1 and S2 are sentences, S1 $\Leftrightarrow$ S2 is a sentence (biconditional)

# Propositional logic: Semantics



- Each model specifies true/false for each **proposition** symbol

  For example P1,2 P2,2 P3,1

  false true false

- With these symbols, 8 possible models, can be enumerated automatically

Rules for evaluating truth with respect to a model m:

| | | | | |
|---|---|---|---|---|
| $\neg S$ | is true iff | S is false | | |
| $S1 \wedge S2$ | is true iff | S1 is true | and | S2 is true |
| $S1 \vee S2$ | is true iff | S1is true | or | S2 is true |
| $S1 \rightarrow S2$ | is true iff | S1 is false | or | S2 is true |
| i.e., | is false iff | S1 is true | and | S2 is false |
| $S1 \Leftrightarrow S2$ | is true iff | S1→S2 is true and S2→S1 is true | | |

- A simple recursive process evaluates an arbitrary sentence, as for example:

  $\neg P1,2 \wedge (P2,2 \vee P3,1) = true \wedge (true \vee false) = true \wedge true = true$

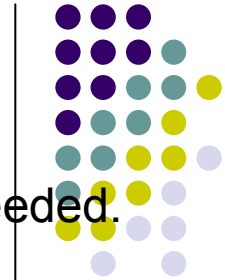
# Truth Tables for Connectives

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| $false$ | $false$ | $true$ | $false$ | $false$ | $true$ | $true$ |
| $false$ | $true$ | $true$ | $false$ | $true$ | $true$ | $false$ |
| $true$ | $false$ | $false$ | $false$ | $true$ | $false$ | $false$ |
| $true$ | $true$ | $false$ | $true$ | $true$ | $true$ | $true$ |

# Logical Equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

# Rules of Inference



- To derive true formulas from other true formulas, rules of inference are needed.
- In a sound theory, the rules of inference preserve truth.
- If all formulas in the starting set are true, only true formulas can be inferred from them.
- Some of the rules of inference for the propositional calculus are as follows:

Let symbols p, q and r represent any formula :

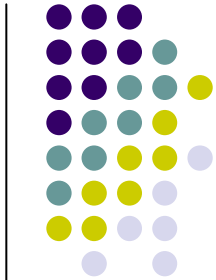
| | | |
|---|---|---|
| **Modus Ponens.** | From **p** and **p** → **q**, | derive **q**. |
| **Modus Tollens.** | From ¬ **q** and **p** → **q**, | derive ¬ **p** |
| **Hypothetical Syllogism.** | From **p** → **q** and **q** → **r**, | derive **p** → **r**. |
| **Disjunctive Syllogism.** | From **p** ∨ **q** and ¬ **p**, | derive **q**. |
| **Conjunction.** | From **p** and **q**, | derive **p** ∧ **q**. |
| **Addition.** | From **p**, | derive **p** ∨ **q** . |

this rule allows any formula to be added to a disjunction

| | | |
|---|---|---|
| **Subtraction.** | From **p** ∧ **q**, | derive **p**. |

this rule simplifies formulas by throwing away unneeded conjuncts

# Limits of Propositional Logic

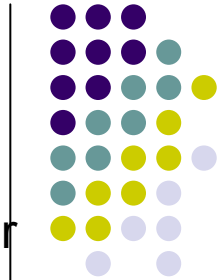


- Propositional logic is declarative
- Propositional logic is compositional:
  - meaning of B1,1 $\wedge$ P1,2 is derived from meaning of B1,1 and of P1,2
- Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- Limits of Propositional logic
  - Propositional logic is not powerful enough as a general knowledge representation language.
  - Impossible to make general statements.
    - Example “all students take exams” or
    - “if any student take an exam, s/he either pass or fail”
- What do we need?

# Exercise 1:

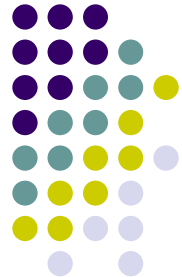
- Demonstrate that $p \rightarrow q$ is equivalent to $\neg (p \wedge \neg q)$
- Prove: $q \rightarrow (p \vee q)$

# Predicate Logic (Introduction)



- Whereas propositional logic assumes the world contains facts, first-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of,
- In predicate logic the basic unit is a predicate/ argument structure called an atomic sentence:
  - likes(alison, chocolate)
  - tall(fred)
- Arguments can be any of:
  - constant symbol, such as ‘alison’
  - variable symbol, such as X
  - A function, such as sqrt(n)
- Examples:
  - Likes(X, richard)
  - Friends(joe, jim)

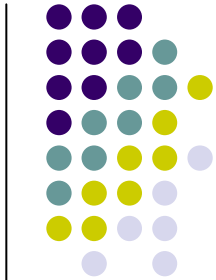
# Predicate Logic (Syntax)



- These atomic sentences can be combined using logic connectives
  - Likes(john, mary) $\wedge$ Tall(mary)
  - Tall(john) $\rightarrow$ Nice(john)
- Sentences can also be formed using quantifiers $\forall$ (forall) and $\exists$ (there exists) to indicate how to treat variables:
  - $\forall$ X ( Lovely(X) )  Everything is lovely.
  - $\exists$ X ( Lovely(X) )  Something is lovely.
  - $\forall$ X ( In(X, garden) $\rightarrow$Lovely(X) )  Everything in the garden is lovely.
- We can have several quantifiers in an expression, such as:
  - $\forall$ X  $\exists$ Y ( Loves(X, Y) )
  - $\forall$ X ( Handsome(X) $\rightarrow$  $\exists$ Y ( Loves(Y, X) ) )

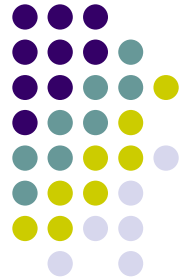
- Here are identities common in predicate calculus:
  - $\exists$ X (P(X)) is identical to $\neg$ $\forall$ x ($\neg$ P(X) )
  - $\forall$ x (P(X)) is identical to $\neg$ $\exists$ x ($\neg$ P(X))

# Predicate Logic (Syntax details)



- Constants KingJohn, 2, Laval,...
- Predicates Brother, Child,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg, \rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality =
- Quantifiers $\forall, \exists$
- Atomic sentence = predicate (term1,...,termn)
  or term1 = term2
- Term = function (term1,...,termn)
  or constant or variable
- Complex sentences are made from atomic sentences using connectives
  $\neg S$, S1 $\wedge$ S2, S1 $\vee$ S2, S1 $\rightarrow$ S2, S1 $\Leftrightarrow$ S2,

# Predicate Logic Semantics



- Sentences are true with respect to a model and an interpretation
- A model contains objects (domain elements) and relations among them
- An interpretation specifies referents for

  constant symbols → objects

  predicate symbols → relations

  function symbols → functional relations
- An atomic sentence Predicate($term_1$,...,$term_n$) is true

  iff the objects referred to by $term_1$,...,$term_n$

  are in the relation referred to by Predicate
- ∀ X P(X) means that P(X) must be true for every object X in the domain of interest
  - ∀x P is true in a model M iff P is true with x being each possible object in the model
- ∃ X P(X) means that P(X) must be true for at least one object X in the domain of interest
  - ∃x P is true in a model M iff P is true with x being some possible object in the model
- Example: if we have a domain of interest consisting of just two people, john and mary, and we know that Tall(mary) and Tall(john) are true, we can say that ∀ X tall(X) is true

# Proof and inference

- We can define inference rules allowing us to say that if certain things are true, certain other things are sure to be true, e.g.

  $\forall$ X (P(X) $\rightarrow$Q(X))
  P(aa)
  ----------  (so we can conclude)
  Q(aa)

- This involves matching P(X) against P(aa) and binding the variable X to the symbol aa

- Example: What can we conclude from the following?

  $\forall$ X Tall(X) $\rightarrow$ Strong(X)

  Tall(john)

  $\forall$ X Strong(X) $\rightarrow$ Loves(mary, X)

- Exercise 2: Demonstrate the equivalence of the two formulas

  $\forall$ x (PEACH(x) $\rightarrow$ FUZZY(x))

  $\neg \exists$ x(PEACH(x) $\wedge \neg$ FUZZY(x))

# Order of Quantifiers

- The order of quantifiers in symbolic logic makes a crucial difference, as it does in English.
- Consider the sentence '*Every man in department C99 married a woman who came from Boston*', which may be represented by the formula:

  $\forall x \exists y ( ((MAN(x) \wedge DEPT(x, C99)) \rightarrow (WOMAN(y) \wedge HOMETOWN(y, BOSTON) \wedge MARRIED(x, y))))$.

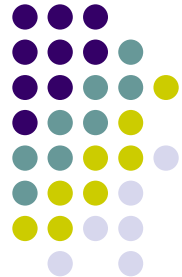
  *The above formula says that for every x there exists a y such that, if x is a man and x works in department C99, then y is a woman, the home town of y is Boston, and x married y.*
- Interchanging the two quantifiers leads to the formula:

  $\exists y \forall x (((MAN(x) \wedge DEPT(x, C99)) \rightarrow (WOMAN(y) \wedge HOMETOWN(y, BOSTON) \wedge MARRIED(x, y))))$.

  *This formula says that there exists a y such that for every x, if x is a man and x works in department C99, then y is a woman, the home town of y is Boston, and x married y.*

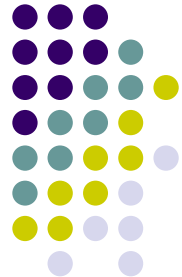
  In English, that would be the same as saying, A woman who came from Boston married every man in department C99!!

# Creation of a FOL Knowledge base (Exercise1/4)



- Goal: we want to specify a FOL KB for Family relationships
- Here are some facts:
  - John is a man, Mary is a woman, Jack is a man, Paul is a man
  - John is the father of Mary
  - John is the son of Mary
  - Paul is the brother of Jack
  - Paul is the son of Mary
- Exercise 3.1:
  - What do you think about these facts?
  - Represent these facts using a predicate notation
- We need to use unary and binary predicates: why?
- What do binary predicates represent?
- Represent this knowledge base as a graph (semantic network)
  - Which FOL elements correspond to the nodes?
  - Which FOL elements correspond to the edges?
- Propose a data conceptual model (DCM) to organize these facts in a data base
- What differences do you notice between the graph form and the DCM?

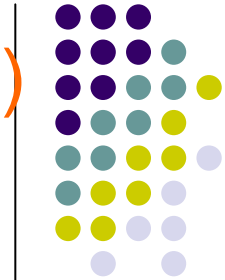
# Creation of a FOL Knowledge base (Exercise2/4)



- In the list of facts there is a contradiction. How can you demonstrate that these facts are contradictory using FOL?
- We need to introduce generic definitions for family relationships such as
  - FatherOf, SonOf, BrotherOf
- Exercise 3.2: Propose FOL formulas for these relationships:

  FatherOf(x,y), SonOf(x,y), BrotherOf(x,y)
  - Are the proposed definitions generic enough?
- We need to identify primitives to represent the knowledge in our Family KB:
  - Which primitives do you choose? Why?
- Choosing primitives (for concepts and relations) which are appropriate for a given domain is the first step of a knowledge representation approach
- We will also call this set of concepts and relations, a base ontology
- Notion of instance:
  - What is the difference between FatherOf(john, mary) and FatherOf(x,y)?
- Using the primitives of the domain ontology, we can specify all the other needed definitions using our FOL formalism. Can you illustrate this using our example?
  - How can we define the relationship GrandFatherOf ?

# Creation of a FOL Knowledge base (Exercise3/4)

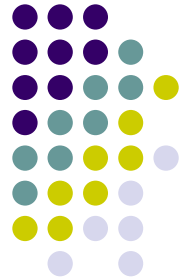


- Let us go back to the contradiction in our example: how can we prove it using our new generic definitions ?
- We need to introduce new definitions which have not been made explicit yet, because as we know 'too' well the domain, we take them for granted.
- This is a major issue when we work with domain experts: how to make them specify all the pieces of knowledge that a reasoning system will need?
- Very often, the additional knowledge that needs to be made explicit corresponds to domain constraints, i.e. formulas (or rules) that need to be respected by all the sentences in the KB so that it remains consistent (i.e. without any contradiction)
- There is a contradiction in a KB whenever we find a contradictory sentence, i.e. a sentence and its negation
- Exercise3.3: How to specify that Parent(x,y) and Parent(y,x) are contradictory?
  - You may need to use negation ($\neg$ ) and the logical equivalences

    $( P \rightarrow Q ) \equiv \neg (P \wedge \neg Q)$

    $\neg \neg P \equiv P$
- Exercise 3.4:
  - Propose a definition for GrandFatherOf(x,y) as 'the father of the mother of y or of the father of y' (using the primitives FatherOf and MotherOf)
  - Propose a definition for GrandFatherOf(x,y) using the primitive ParentOf and what else?

# Creation of a FOL Knowledge base (Exercise4/4)



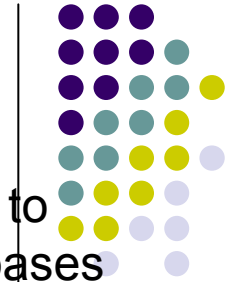
- Exercise 3.5: (be careful with the use of quantifiers)
  - Propose a definition for GrandParentOf(x,y)
  - Propose a definition for GreatGrandParentOf(x,y)
  - Propose a definition for AncesterOf
    - We may need to use a parameter to specify the level of ancester you are considering (recursive definition)
- Synthesis:
  - Importance of choosing the right primitives for a given domain
  - Importance of providing the right definitions based on these primitives to define the concepts and relations relevant for the given domain
  - Notion of ontology
  - Notice also that when using predicates, the order of the arguments conveys meaning

    Example:

    Gives (john, mary, flowers) which is different from Gives (john, mary, flowers)

    ↑Agnt, ↑Rec. , ↑Obj ↑Rec. , ↑Agnt, ↑Obj

# Introduction to Reasoning with Expert Systems



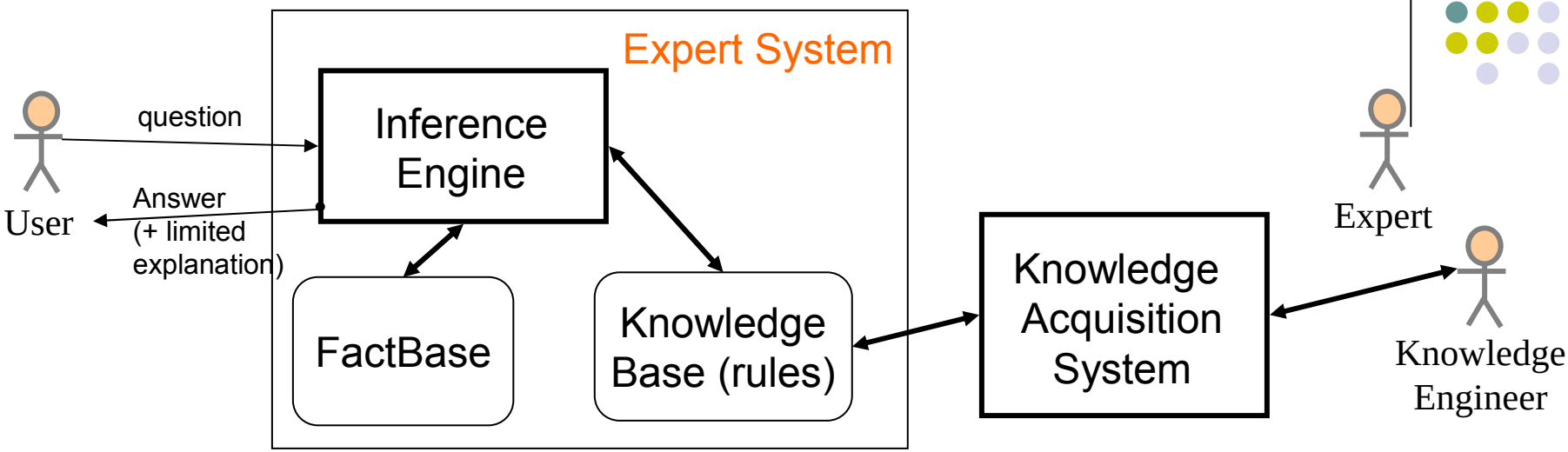
- Expert systems have been developed in the 1970s as practical systems to reason on knowledge expressed in terms of rules and facts as flat databases (triplets). In the 1980s other knowledge representations have been used with rules, i.e. semantic nets, frames, etc.
- An expert system is an automatic reasoner which is based on the logic inference rule called *Modus Ponens*:

| | | | |
|---|---|---|---|
| $\forall$ X (P(X) $\rightarrow$Q(X)) | | IF P Then Q | |
| P(aa) | | P | |
| ---------- | or | ---------- | (inference) |
| Q(aa) | | Q | |

- In logic there is another well-known inference rule called *Modus Tollens*

| | | | |
|---|---|---|---|
| $\forall$ X (P(X) $\rightarrow$Q(X) ) | | IF P Then Q | |
| $\neg$ Q(aa) | | $\neg$ Q | |
| ---------- | or | ---------- | |
| $\neg$ P(aa) | | $\neg$ P | |

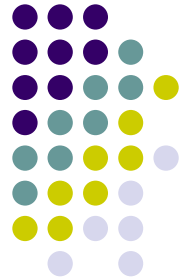
# Fundamental Characteristics of Expert Systems



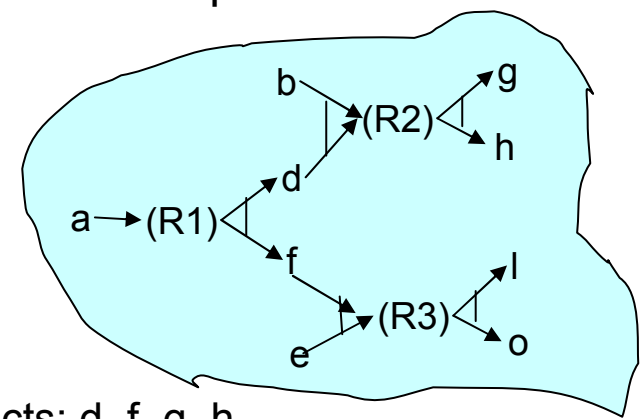

- The inference engine is the reasoning module which uses *Modus Ponens*

  IF P(x) Then Q(x) rule
  P(a) fact
  ---------- (inference)
  Q(a) deduced fact
- The inference engine matches facts (P(a)) and rule premises (IF P(x)) to deduce new facts Q(a)
- It also chains rules: IF P Then Q and IF Q Then R

# Forward and Backward Chaining (1/2)

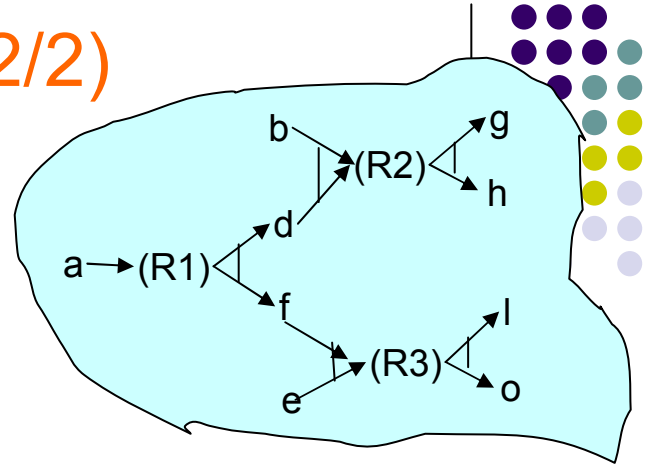


- There are two ways of using the inference engine called forward-chaining and backward-chaining
- *Forward-chaining* consists in starting from facts describing a situation and using the rule base to try to deduce as many new facts as it is possible (saturation of the fact base). This is a direct use of the modus ponens inference rule
- Example (Forward Chaining)
  - Facts: b, c, m, n
  - Rules: R1 IF a Then d And f
    R2 IF b And d Then g And h
    R3 IF f And e Then l And o



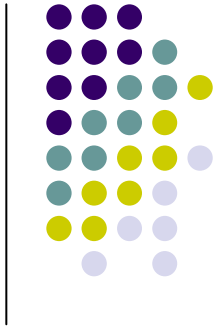

  - New fact1 (provided by user): a Deduced facts: d, f, g, h
  - New fact2 (provided by user): e Deduced facts: none
  - New fact3 (provided by user): a And e Deduced facts: d, f, g, h, l, o
- *Backward-chaining* consists in setting an hypothetical fact (in Prolog terms we speak of a goal) and using the rule base and the inference engine to go backward and to try to retrieve the facts in the fact base and the chain of rules that enable to deduce the hypothetical fact

# Forward and Backward Chaining (2/2)




- Example (Backward Chaining)
  - Facts: a, b, c, m, n
  - Rules: R1 IF a Then d And f
    R2 IF b And d Then g And h
    R3 IF f And e Then l And o
  - hypothesis (submitted by user): f
    Proof: No
    Explanation: R1 cannot be triggered
    Advice: Try to verify fact a
  - New fact (provided by user): a
    Proof: f true (backward chaining)
    Explanation: d And f true because rule R1 and fact a
    Other deduced facts: g, h
- Notice that Backward Chaining is used to prove that an hypothetical fact is true according to the current contents of the fact and knowledge base.
- Backward Chaining can also be used to identify facts that need to be verified or observed in the real world so that we can prove an hypothesis
- In practice, most systems use both Forward and Backward Chainings

# Example in the Medical Domain



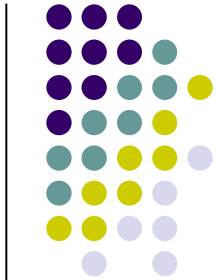
- Example: A patient consults a physician for some illness,
  - The physician asks the patient for data
    - symptoms and patient's characteristics → Facts
  - The physician uses his/her medical knowledge (rules about symptoms and illnesses) to deduce new facts (forward-chaining)
  - He can also set hypothesis (backward chaining). Based on an hypothesis ('patient has the flu'), the physician may ask questions to the patient to verify facts that have not been identified yet, or the physician may ask the patient to undergo tests to verify some facts (Symptoms). With all these facts the physician can deduce the patient's illness (forward-chaining) with a certain degree of certainty
  - When the physician has identified the patient's illness, s/he uses his/her knowledge of cures (rules about illnesses and medication) to prescribe some medecine to the patient (forward-chaining)
- Exercise 4:
  - Let us consider the Family KB

    Man(john), Man(jack), Man(peter), Man(paul), Woman(mary)

    Parent(john, mary) Parent(mary, peter)

    Parent(mary, paul) Brother(paul, jack)
  - Can you write an expression to find the Brother(s) of Paul
    - Hint: write and test predicate Brother (xpaul) ? Which type of reasoning are you using?

# About Data Bases and Expert Systems



- In an expert system, the fact base corresponds to a simple data base
- In the relational data model, T. Codd used predicate calculus to mathematically justify the mechanisms used by relational data base management systems
  - Hence, the notion of table columns which are the arguments of predicates and table rows which are instances of predicates in the data base
- A data base equipped with an inference engine (called a deductive data base) is able to infer new facts automatically, but also to validate constraints and coherence of new data with respect to the current content of the data base
- Closed world hypothesis (Hypothèse du monde clos)
  - According to this hypothesis, all data used to carry out deductions or to answer queries must be recorded in the data base (or fact base) used by the system
  - Any information which is not recorded in the data base (or fact base) is considered as being false (or unknown) by the system
- Hence, the data base (or fact base) is considered as the *World of reference* by the system
- Exercise:
  - Given the predicates and facts of our Family KB, can you illustrate the closed-world hypothesis?