SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – Introduction to Artificial Intelligence

II Year / III Semester

Unit 2 Rule based deduction system

Rule-Based Deduction Systems

The way in which a piece of knowledge is expressed by a human expert carries important information,

example: if the person has fever and feels tummy-pain then she may have an infection.

In logic it can be expressed as follows:

$\forall x. (has\_fever(x) \; \& \; tummy\_pain(x) \rightarrow has\_an\_infection(x))$

If we convert this formula to clausal form we loose the content as then we may have equivalent formulas like:

(i) $has\_fever(x) \; \& \; \sim has\_an\_infection(x) \rightarrow \sim tummy\_pain(x)$

(ii) $\sim has\_an\_infection(x) \; \& \; tummy\_pain(x) \rightarrow \sim has\_fever(x)$

Notice that:

- (i) and (ii) are logically equivalent to the original sentence
- they have lost the main information contained in its formulation.

Forward production systems

- The main idea behind the forward/backward production systems is:
  - to take advantage of the implicational form in which production rules are stated by the expert
  - and use that information to help achieving the goal.
- In the present systems the formulas have two forms:
  - rules
  - and facts

Forward production systems

- Rules are the productions stated in implication form.
  - Rules express specific knowledge about the problem.
  - Facts are assertions not expressed as implications.
  - The task of the system will be to prove a goal formula with these facts and rules.
  - In a forward production system the rules are expressed as *F-rules*
  - F-rules operate on the global database of facts until the termination condition is achieved.
  - This sort of proving system is a *direct* system rather than a refutation system.
- Facts
  - Facts are expressed in AND/OR form.
  - An expression in AND/OR form consists on sub-expressions of literals connected by & and V symbols.
  - An expression in AND/OR form is not in clausal form.

Forward production systems

Steps to transform facts into AND/OR form for forward system:

1. *Eliminate* (temporarily) implication symbols.
2. Reverse quantification of variables in first disjunct by *moving negation symbol.*
3. *Skolemize* existential variables.
4. Move all ***universal quantifiers*** to the front an drop.
5. ***Rename variables*** so the same variable does not occur in different main conjuncts
   - Main conjuncts are small AND/OR trees, not necessarily sum of literal clauses as in Prolog.

EXAMPLE

Original formula: ∃u. ∀v. {q(v, u) & ~[[r(v) v p(v)] & s(u,v)]}

converted formula: q(w, a) & {[~r(v) & ~p(v)] v ~s(a,v)}

Conjunction of two main conjuncts

Various variables in conjuncts

All variables appearing on the final expressions are assumed to be universally quantified.

...Based Deduction Systems: forward production systems

...ules

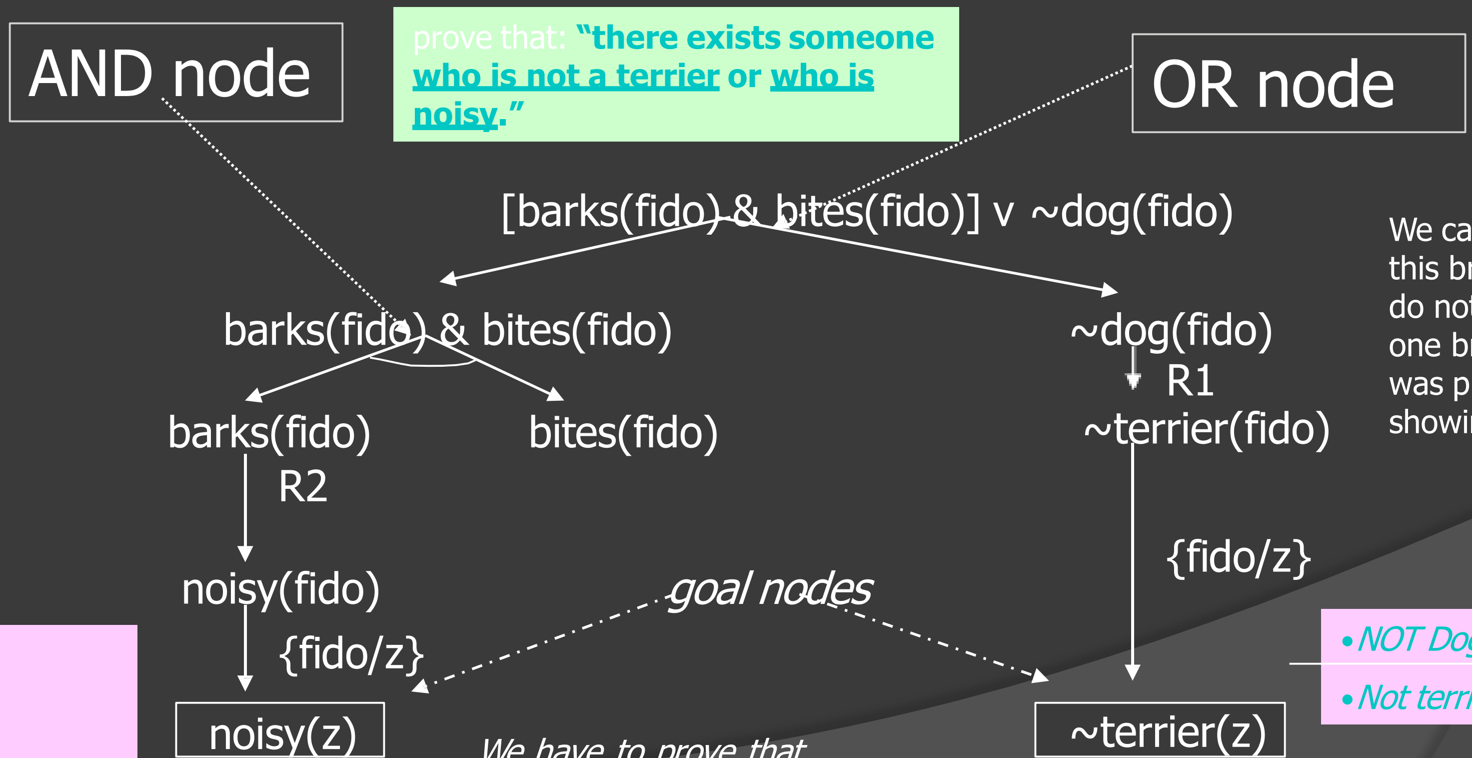
Rules in a forward production system will be applied to the AND/OR graph to produce new transformed graph structures.

We assume that rules in a forward production system are of the form:

L ==> W,

where L is a literal and W is a formula in AND/OR form.

- Recall that a rule of the form (L1 V L2) ==> W is equivalent to the pair of rules: L1 ==> W V L2 ==> W.



We cannot prove this branch but we do not have to since one branch of OR was proven by showing Fido

- *NOT Dog(Fido)*
- *Not terrier(Fido)|*

- *Dog(Fido)*
- *barks(Fido)*
- *Not terrier(Fido)|*
- *Noisy(Fido)*

We have to prove that there is X that is noisy.
X=Fido

Or we have to prove that there is X that X is not a terrier

forward production systems

<u>Steps to transform the rules into a free-quantifier form:</u>

1. Eliminate (temporarily) implication symbols.
2. Reverse quantification of variables in first disjunct by moving negation symbol.
3. Skolemize existential variables.
4. Move all universal quantifiers to the front and drop.
5. Restore implication.

All variables appearing on the final expressions are assumed to be universally quantified.

E.g. Original formula: $\forall x.(\exists y.\ \forall z.\ (p(x, y, z)) \rightarrow \forall u.\ q(x, u))$

Converted formula: $p(x, y, f(x, y)) \rightarrow q(x, u).$

Skolem function

Restored implication

forward production systems

A full ***example:***

- Fact: Fido barks and bites, or Fido is not a dog.
- (R1) All terriers are dogs.
- (R2) Anyone who barks is noisy.

Based on these facts, prove that: **"there exists someone who is not a terrier or who is noisy."**

goal

Logic representation:

(barks(fido) & bites(fido)) v ~dog(fido)

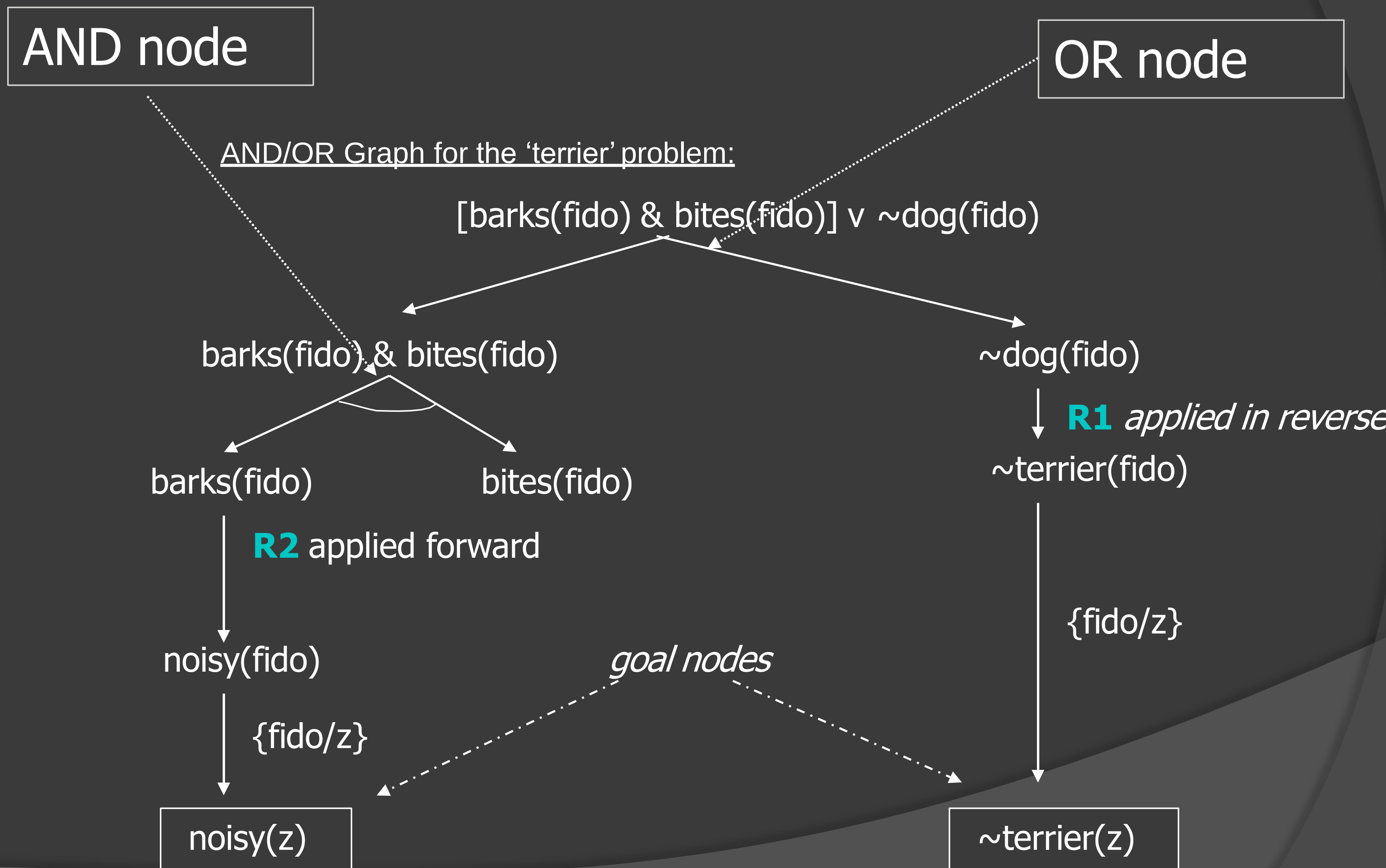
R1: terrier(x) → dog(x)

R2: barks(y) → noisy(y)

goal: ∃w.(~terrier(w) v noisy(w))

-Based Deduction Systems: forward production syste

From facts to goal



Backward production systems

B-Rules

We restrict B-rules to expressions of the form: W ==> L,

where W is an expression in AND/OR form and L is a literal,

and the scope of quantification of any variables in the implication is the entire implication.

Recall that W==>(L1 & L2) is equivalent to the two rules: W==>L1 and W==>L2.

An important property of logic is the duality between assertions and goals in theorem-proving systems.

Duality between assertions and goals allows the goal expression to be treated as if it were an assertion.

Conversion of the goal expression into AND/OR form:

1. Elimination of implication symbols.
2. Move negation symbols in.
3. Skolemize *existential* variables.
4. Drop existential quantifiers. Variables remaining in the AND/OR form are considered to be existentially quantified.

Goal clauses are conjunctions of literals and the disjunction of these clauses is the clause form of the goal well-formed formula.

ample 1 of formulation of Rule-Based Deduction Sys

1. Facts:
   - dog(fido)
   - ~barks(fido)
   - wags-tail(fido)
   - meows(myrtle)

Rules:

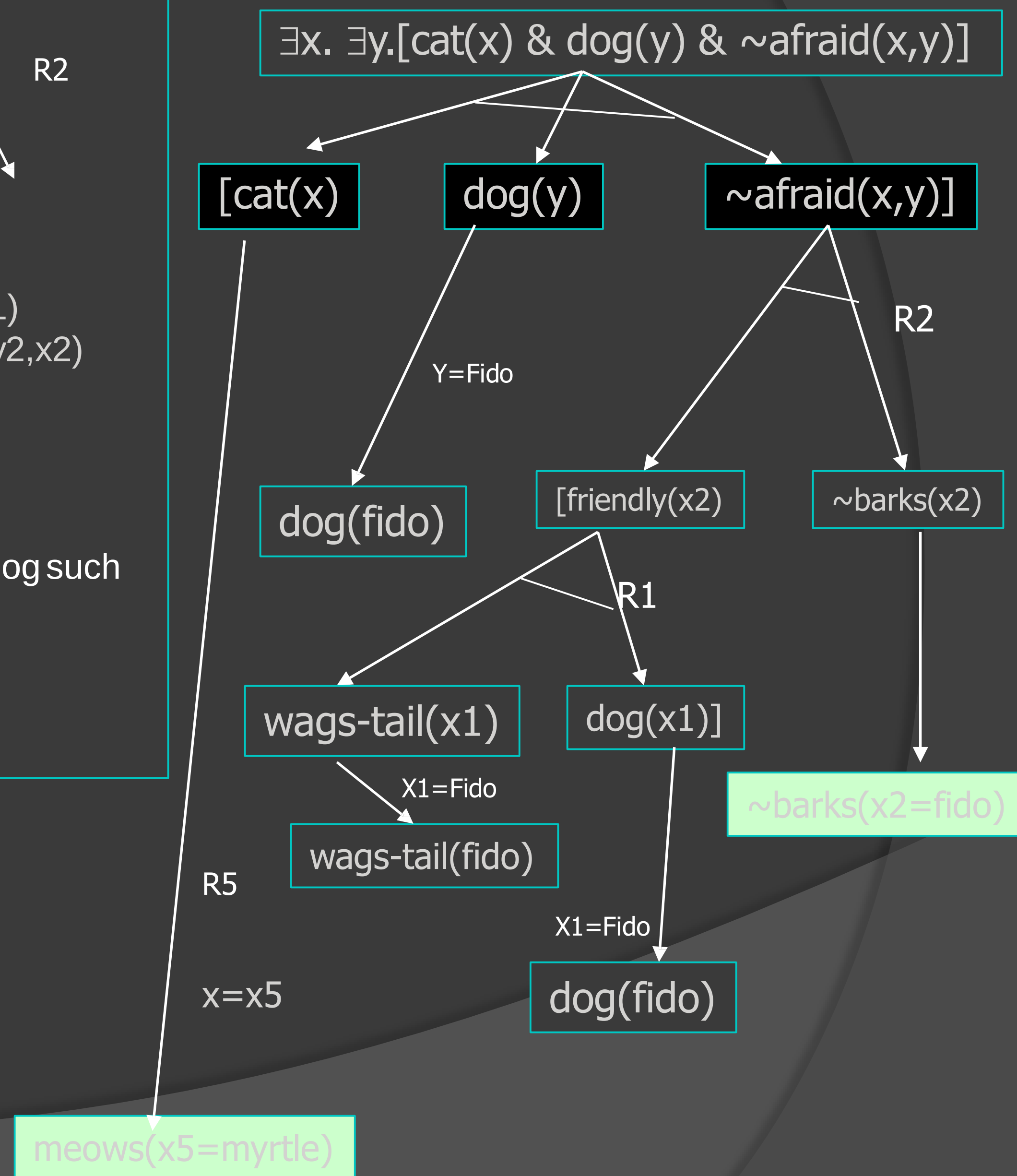
- R1: [wags-tail(x1) & dog(x1)] → friendly(x1)
- R2: [friendly(x2) & ~barks(x2)] → ~afraid(y2,x2)
- R3: dog(x3) → animal(x3)
- R4: cat(x4) → animal(x4)
- R5: meows(x5) → cat(x5)

Suppose we want to ask if there are a cat and a dog such that the cat is unafraid of the dog.

The goal expression is:

$\exists x. \exists y.[cat(x) \;\&\; dog(y) \;\&\; \sim afraid(x,y)]$

We treat the goal expression as an assertion



mework: formulation of Rule-Based Deduction S

2. The blocks-word situation is described by the following set of wffs:

on_table(a) clear(e)

on_table(c) clear(d)

on(d,c) heavy(d)

on(b,a) wooden(b)

heavy(b) on(e,b)

The following statements provide general knowledge about this blocks word:

Every big, blue block is on a green block.

Each heavy, wooden block is big.

All blocks with clear tops are blue.

All wooden blocks are blue.

Represent these statements by a set of implications having single-literal consequents.

Draw a consistent AND/OR solution tree (using B-rules) that solves the problem: “Which block is on a green block?”

HOMEWORK Problem 2. Transformation of rules and goal:

Facts:

f1: on_table(a)	f6: clear(e)
f2: on_table(c)	f7: clear(d)
f3: on(d,c)	f8: heavy(d)
f4: on(b,a)	f9: wooden(b)
f5: heavy(b)	f10: on(e,b)

Rules:

R1:	big(y1) ^ blue(y1) → green(g(y1))	Every big, blue block is on a green block.
R2:	big(y0) ^ blue(y0) → on(y0,g(y0))	“ “ “ “ “ “ “ “ “
R3:	heavy(z) ^ wooden(z) → big(z)	Each heavy, wooden block is big.
R4:	clear(x) → blue(x)	All blocks with clear tops are blue.
R5:	wooden(w) → blue(w)	All wooden blocks are blue.

Goal:

green(u) ^ on(v,u) Which block is on a green block?

HOMEWORK PROBLEM 3. Information Retrieval System

- We have a set of facts containing personnel data for a business organization
- and we want an automatic system to answer various questions about personal matters.
- Facts

John Jones is the manager of the Purchasing Department.

manager(p-d,john-jones)

works_in(p-d, joe-smith)

works_in(p-d,sally-jones)

works_in(p-d,pete-swanson)

Harry Turner is the manager of the Sales Department.

manager(s-d,harry-turner)

works_in(s-d,mary-jones)

works_in(s-d,bill-white)

married(john-jones,mary-jones)