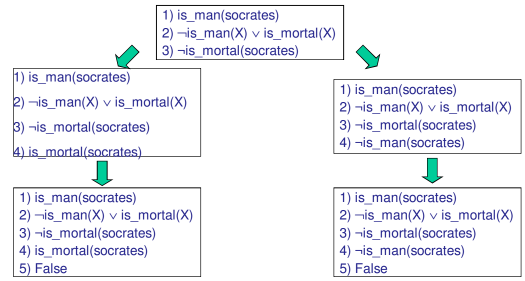
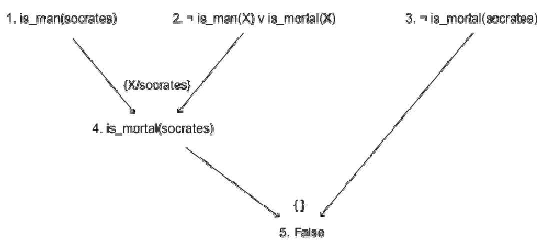
# Aristotle's Example

- Socrates is a man and all men are mortal
  Therefore Socrates is mortal
- Initial state
  1) is_man(socrates)
  2) ¬is_man(X) ∨ is_mortal(X)
  3) ¬is_mortal(socrates) (negation of theorem)
- Resolving (1) & (2) gives new state
  (1)-(3) & 4) is_mortal(socrates)

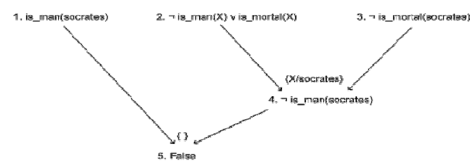
# Aristotle's Example: Search Space




# Resolution Proof Tree (Proof 1)




# Resolution Proof Tree (Proof 2)




You said that all men were mortal. That means that for all things X, either X is not a man, or X is mortal. If we assume that Socrates is not mortal, then, given your previous statement, this means Socrates is not a man. But you said that Socrates *is* a man, which means that our assumption was false, so Socrates must be mortal.

# Example: KB

Jack owns a dog.
Every dog owner is an animal lover.
No animal lover kills an animal.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

# Example: KB

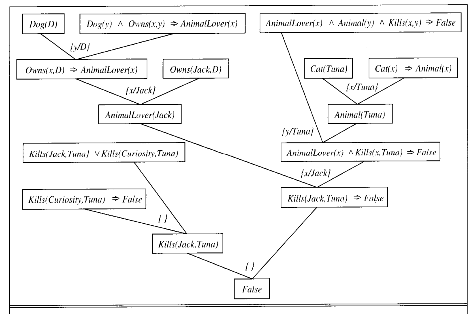
Jack owns a dog.
Every dog owner is an animal lover.
No animal lover kills an animal.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

A. $\exists x \; Dog(x) \wedge Owns(Jack, x)$
B. $\forall x \; (\exists y \; Dog(y) \wedge Owns(x, y)) \Rightarrow AnimalLover(x)$
C. $\forall x \; AnimalLover(x) \Rightarrow \forall y \; Animal(y) \Rightarrow \neg Kills(x, y)$
D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $\forall x \; Cat(x) \Rightarrow Animal(x)$

# Example: (CNF)

A1. $Dog(D)$
A2. $Owns(Jack, D)$
B. $Dog(y) \wedge Owns(x, y) \Rightarrow AnimalLover(x)$
C. $AnimalLover(x) \wedge Animal(y) \wedge Kills(x, y) \Rightarrow False$
D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $Cat(x) \Rightarrow Animal(x)$

# Example: Proof Tree




# Reduction to propositional inference

Suppose the KB contains just the following:

∀x King(x) ∧ Greedy(x) ⇒ Evil(x)

King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:

King(John) ∧ Greedy(John) ⇒ Evil(John)
King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)
King(John)
Greedy(John)
Brother(Richard,John)

- The new KB is propositionalized: proposition symbols are
- King(John), Greedy(John), Evil(John), King(Richard), etc.

# Reduction to propositional inference

- Every FOL KB can be propositionalized so as to preserve entailment (A ground sentence is entailed by new KB iff entailed by original KB)
- Idea: propositionalize KB and query, apply resolution in PC, return result
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*)))

# Reduction to propositional inference

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

**Problem**: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is

semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

# Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from:
- ∀x King(x) ∧ Greedy(x) ⇒ Evil(x)

King(John)
∀y Greedy(y)
Brother(Richard,John)

- it seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant

## Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

$p_1'$ is *King*(*John*) $p_1$ is *King*(*x*)

$p_2'$ is *Greedy*(*y*) $p_2$ is *Greedy*(*x*)

θ is {x/John,y/John} q is *Evil*(*x*)

q θ is *Evil*(*John*)

## Soundness and Completeness of GMP

- GMP is sound
  - Only derives sentences that are logically entailed
- GMP is complete for a KB consisting of definite clauses
  - Complete: derives all sentences that are entailed
  - OR…answers every query whose answers are entailed by such a KB
  - Definite clause: disjunction of literals of which exactly one is positive,

## Forward chaining

- FC: "Idea" fire any rule whose premises are satisfied in the *KB*, add its conclusion to the *KB*, until query is found
- Deduce new facts from axioms
- Hopefully end up deducing the theorem statement
- Can take a long time: not using the goal to direct search

## Backward chaining

- BC: "Idea" work backwards from the query *q in (p→q)*

  check if *q* is known already known, or

  prove by BC all premises of some rule concluding *q*
- Start with the conclusion and work backwards
  - Hope to end up at the facts from KB
- Each step asks: given the state that I'm at...
  - Which operator could have been applied to which state to produce the state (sentence) I'm at

Remarks:

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal has already been proved true, or has already failed

## Example

Consider the following knowledge base:

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Prove that Col. West is a criminal
- We will do it through FC, BC, resolution

## Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

*American(x)* ∧ *Weapon(y)* ∧ *Sells(x,y,z)* ∧ *Hostile(z)* ⇒ *Criminal(x)*

¬ *American(x)* ∨ ¬ *Weapon(y)* ∨ ¬ *Sells(x,y,z)* ∨ ¬ *Hostile(z)* ∨ *Criminal(x)*

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

$Owns(Nono,M_1) \wedge Missile(M_1)$

… all of its missiles were sold to it by Colonel West

*Missile(x)* ∧ *Owns(Nono,x)* ⇒ *Sells(West,x,Nono)*

Missiles are weapons:

*Missile(x)* ⇒ *Weapon(x)*

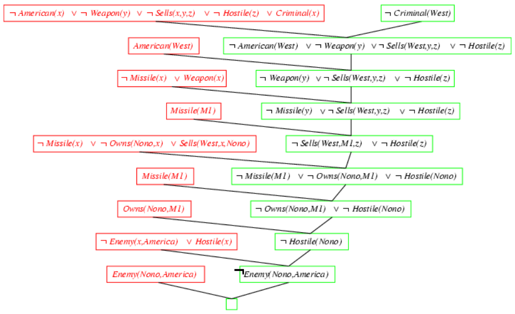
An enemy of America counts as "hostile":

*Enemy(x,America)* ⇒ *Hostile(x)*

West, who is American …*American(West)*

The country Nono, an enemy of America …*Enemy(Nono,America)*

# Resolution proof: definite clauses




---

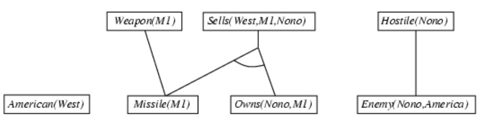
# F C proof

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*


---

# F C proof

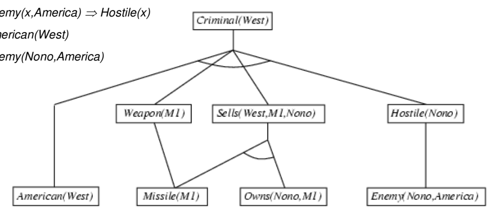
*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*




---

# F C proof

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*




---

# Properties of forward chaining

- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

Forward chaining is widely used in deductive databases

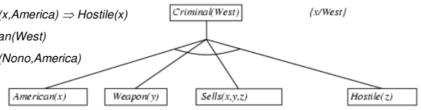
---

# B C proof

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

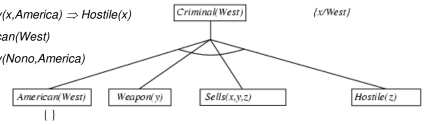
*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof



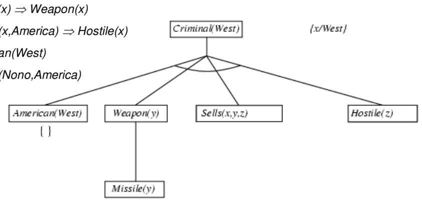

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof



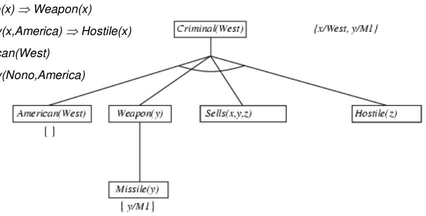

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof



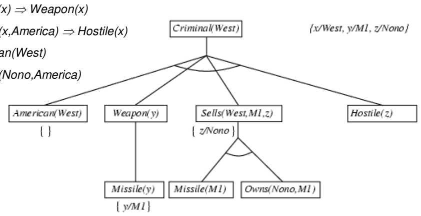

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof



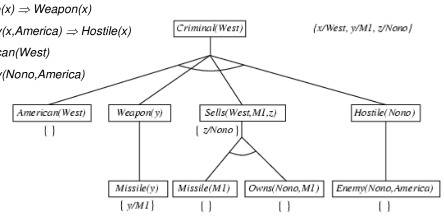

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof




*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
∃x Owns(Nono,x) ∧ Missile(x)
*Owns(Nono,$M_1$) ∧ Missile($M_1$)*
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
*Missile(x) ⇒ Weapon(x)*
*Enemy(x,America) ⇒ Hostile(x)*
*American(West)*
*Enemy(Nono,America)*

## B C proof



## Properties of backward chaining

- Depth-first recursive proof search
- Incomplete due to infinite loops
  - $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$ fix using caching of previous results (extra space)
- Widely used for logic programming
- PROLOG is backward chaining