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

- 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

- Use modus ponens to always derive all consequences from new information
- To avoid looping and duplicated effort, must prevent addition of a sentence to the KB which is the same as one already present.

Problems with Forward Chaining

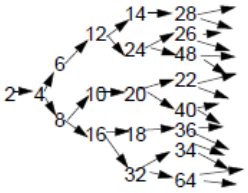
- Inference can explode forward and may never terminate.

Even(x) ➔ Even(plus(x,2))

Integer(x) ➔ Even(times(2,x))

Even(x) ➔ Integer(x)

Even(2)



Forward chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

repeat until *new* is empty
  new ← { }
  for each sentence r **in** KB **do**
    $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
    for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
        for some $p'_1, \ldots, p'_n$ in KB
      $q' \leftarrow$ SUBST(θ, q)
      if q' is not a renaming of a sentence already in KB or *new* **then do**
        add q' to *new*
        $\phi \leftarrow$ UNIFY(q', α)
        if ϕ is not *fail* **then return** ϕ
  add *new* to KB
return *false*

Backward chaining

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

  check if ***q*** is 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
- Widely used for logic programming
- PROLOG is backward chaining

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

Backward Chaining

- Start from a query or atomic sentence to be proven and look for ways to prove it
- Query can contain variables
- Inference process should return all sets of variables hat satisfy the query
- First try to answer query by unifying it to all possible facts in the KB
- Next try to prove it using a rule whose consequent unifies with the query and try to prove all its antecedents recursively

Backward chaining algorithm

function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\}$
  local variables: ans, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each r **in** KB where STANDARDIZE-APART(r) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    $ans \leftarrow$ FOL-BC-ASK($KB, [p_1, \ldots, p_n | \text{REST}(goals)], \text{COMPOSE}(\theta, \theta')) \cup ans$
  return ans

Inference approaches in FOL

- Forward-chaining
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses
- Backward-chaining
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference
- Resolution-based inference (FOL)
  - Refutation-complete for general KB
    - Can be used to confirm or refute a sentence p (but not to generate all entailed sentences)
  - Requires FOL KB to be reduced to CNF
  - Uses generalized version of propositional inference rule

Knowledge Representation Building Knowledge Base in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

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:

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee Criminal(x)$

Nono … has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$ Missile(x):

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

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

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

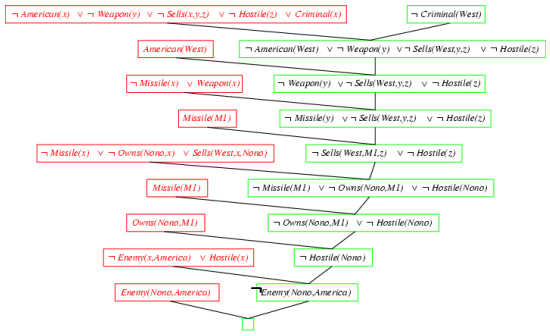
An enemy of America counts as "hostile":

$Enemy(x,America) \Rightarrow Hostile(x)$

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

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

Resolution proof: definite clauses



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)

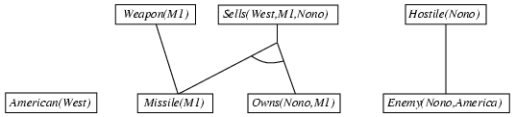
Missile(x) ⇒ Weapon(x)

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)

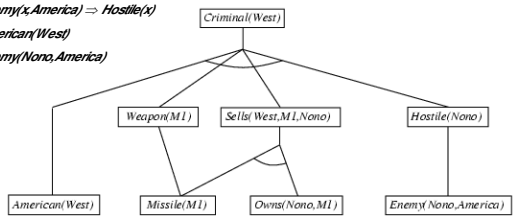
Missile(x) ⇒ Weapon(x)

Enemy(x,America) ⇒ Hostile(x)

American(West)

Enemy(Nono,America)

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)

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)

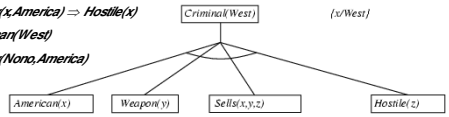
Missile(x) ⇒ Weapon(x)

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)

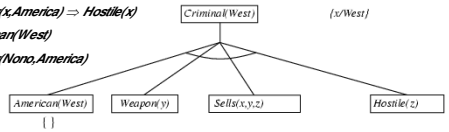
Missile(x) ⇒ Weapon(x)

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)

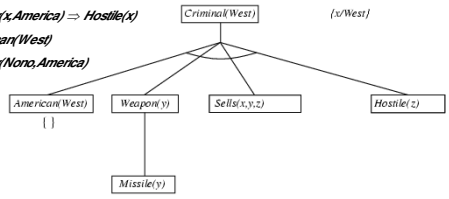
Missile(x) ⇒ Weapon(x)

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)

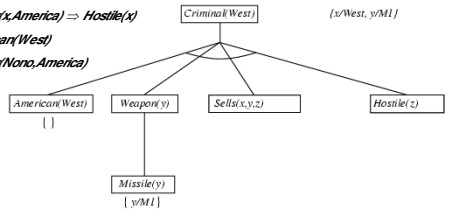
Missile(x) ⇒ Weapon(x)

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)

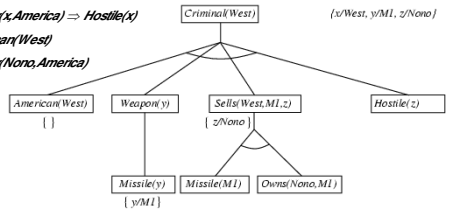
Missile(x) ⇒ Weapon(x)

Enemy(x,America) ⇒ Hostile(x)

American(West)

Enemy(Nono,America)

B C proof



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