4.3 Inference in FOL

SUBST(θ, α) denoteS the result of applying the substitution θ to the sentence α. For example:

SUBST({ x/Sam, y/Pam} , Likes(x, y)) = Likes(Sam, Pam)

The three new inference rules are as follows:

Universal Instantiation	**Existential Instantiation**	**Existential Introduction**
$\dfrac{\forall v \quad \alpha}{\text{SUBST}(\{v/g\},\alpha)}$	$\dfrac{\exists v \quad \alpha}{\text{SUBST}(\{v/k\},\alpha)}$	$\dfrac{\alpha}{\exists v \quad \text{SUBST}(\{g/v\},\alpha)}$
example, from ∀*x Likes(x, IceCream),* we can use the substitution *{x/Ben}* and infer *Likes(Ben, IceCream).*	example from ∃ *x Kill(x, Victim),* we can infer *Kill(Murderer, Victim),* as long as *Murderer* does not appear elsewhere in KB	example, from *Likes(Jerry, IceCream)* we can infer ∃ *x Likes(x, IceCream).*

Generalized Modus Ponens:

For atomic sentences p_i, p_i', and q, where there is a substitution θ such that SUBST(θ, p_i') = SUBST(θ, p_i), for all *i:*

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

Unification UNIFY *(p, q) =* θ where SUBST(θ, *p*) = SUBST(θ, *q*) θ is called the **unifier**

UNIFY (Knows(John, x), Knows(John, Jane)) = {x/Jane}
UNIFY (Knows(John, x), Knows(y, Leonid)) = {x/Leonid, y/John}
UNIFY (Knows(J, x), Knows(y, Mother (y))) = {y/J, x/Mother(J)}
UNIFY (Knows(John, x), Knows(x, Elizabeth)) =fail

An Example Proof

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. What we wish to prove is that **West is a criminal**

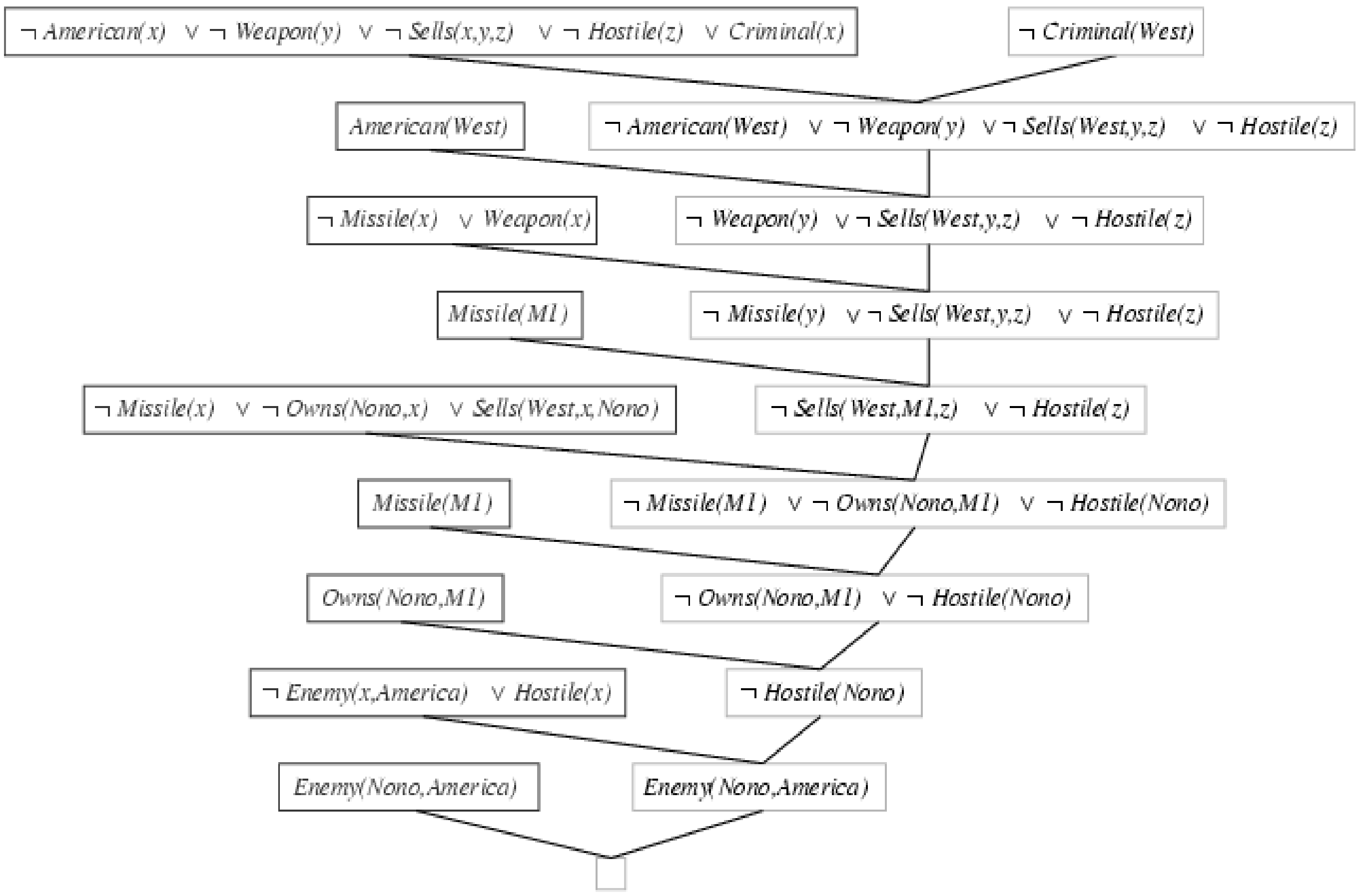
We first represent these facts in first-order logic.

"... it is a crime for an American to sell weapons to hostile nations": ∀*x, y, z American(x)*∧*Weapon(y)*∧*Nation(z)*∧*Hostile(z)*∧*Sells(x,z,y)* ⇒*Criminal(x)*	(1)
"Nono ... has some missiles": ∃ *x Owns(Nono, x)* ∧ *Missile(x)*	(2)
"All of its missiles were sold to it by Colonel West": ∀ *x Owns(Nono, x)*∧*Missile(x)*⇒*Sells(West, Nono, x)*	(3)
We will also need to know that missiles are weapons: ∀ *x Missile(x)* ⇒ *Weapon(x)*	(4)
and that an enemy of America counts as "hostile": ∀ *x Enemy(x, America)*⇒ *Hostile(x)*	(5)
"West, who is American ...": *American(West)*	(6)
"The country Nono ...": *Nation(Nono)*	(7)
"Nono, an enemy of America ...": *Enemy (Nono, America)* *Nation(America)*	(8) (9)

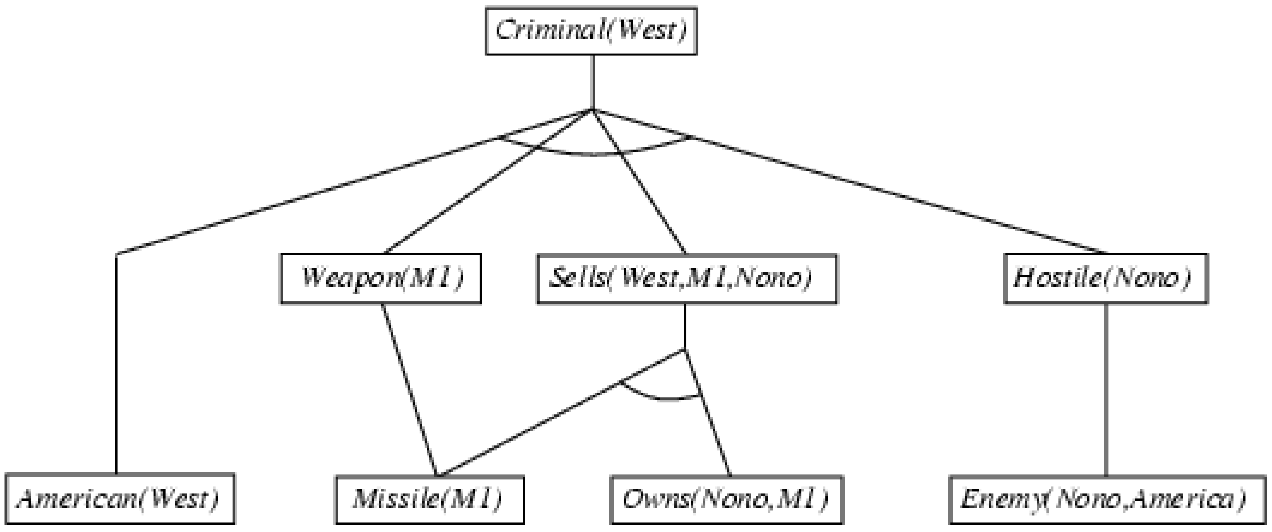
algorithm: convert WFF to clause form

1. eliminate →
2. reduce scope of each ¬ to a single term
3. standardize variables
4. move quantifiers to left of formula (<u>prenex normal form</u>)
5. eliminate existential quantifier
6. drop the prefix
7. convert into conjunction of disjuncts
8. create a separate clause for each conjunct
9. standardize apart the variables

Resolution


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)

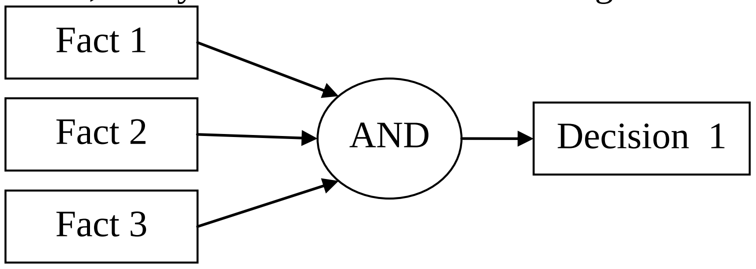
Forward chaining/Backward chaining



Factors that influence question of whether it is better forward or backward

1. Are there more possible start states or goal states?
2. In which direction is the branching factor? We proceed in lower branching factor.
3. Will the program will be asked to justify its reasoning process to a user?
4. What kind of event is going to trigger a problem-solving episode?
   If it is the arrival of a new fact, forward reasoning makes sense.
   If it is a query to which a response is desired, backward reasoning is more natural.

One feature of logic programs that makes it very useful is the justification and explanation of HOW and WHY. The system must be able to explain HOW it arrived a conclusion and WHY it is performing some computation. To answer how a conclusion was reached, work back through the inference chain. To answer why a computation is performed, the system must state its current goal.



Forward reasoning	**Backward reasoning**
It is called data driven reasoning	It is called goal driven reasoning
An inference technique which uses IF THEN rules to reduce a problem solution from initial data.	An inference technique which uses IF THEN rules to repetitively break a goal into smaller sub-goals which are easier to prove.
The system keeps track of current state of problem solution and looks for rules which move state closer to final solution.	The system keeps track of the final state of problem solution and looks for rules which will move that state closer to start state.
Matching is more complex for forward chaining systems	Matching is not complex for backward chaining systems
To reason forward, the left sided are matched against the current state and the right sides (results) are used to generate new nodes until the goal is reached.	To reason backward, the right sides are matched against the current node and the left sides are used to generate new nodes representing new goal states to be achieved. Continue until one of these goal states is matched by initial state.
Examples: Prolog: unification procedure MYCIN: probabilistic certainty factors	Examples: Diagnostic and controlling Natural language understanding