Inference in First Order Logic

Inference in First Order Logic

Inference in PL

- Using rules do not consider quantifiers

Apply inference rules in FOL

- Removing the quantifiers
  - Convert FOL sentences to PL sentences
- Use propositional inference

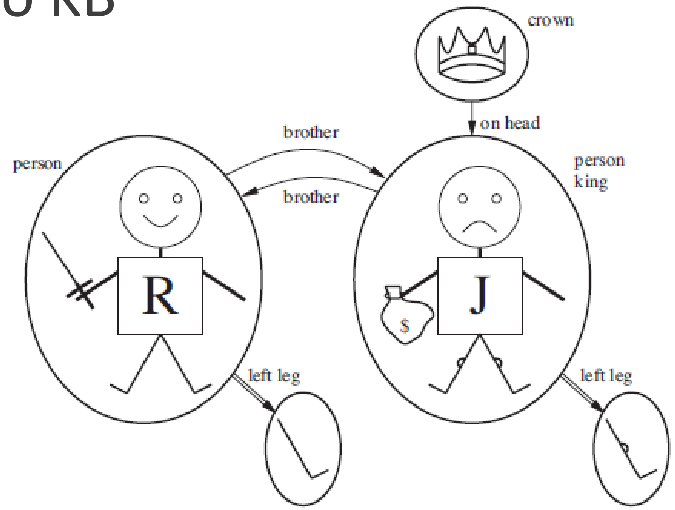
Rules for removing quantifiers

- Instantiation
  - Using all domain elements
  - Give a set of PL sentences

Universal Instantiation (UI)

Substitute **ground term** for the variable

- Assert sentences to KB



$\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

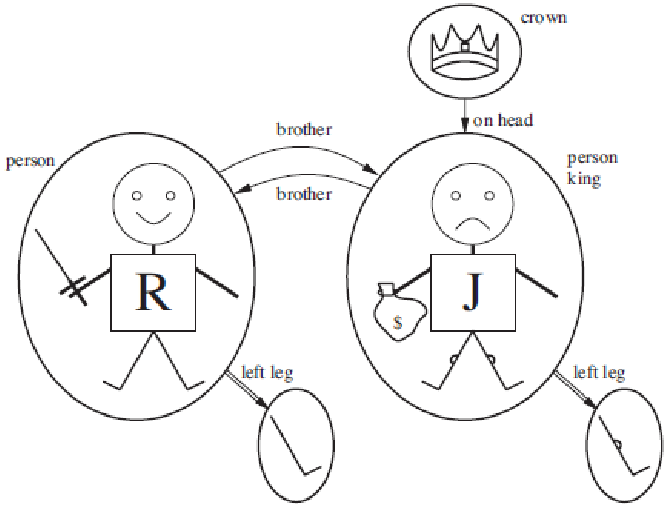
King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
⋮

Universal Instantiation (UI)

SUBST(θ , α) is used to denote

- Result of applying substitution θ to sentence α
- $\theta = \{x/g\}$
  - {x / John}, {x / Richard}
- α is King(x) ∧ Greedy(x) ⇒ Evil(x)

$$\frac{\forall x \;\; \alpha}{\text{SUBST}(\{x/g\}, \alpha)}$$



Existential Instantiation

Remove quantifier again

Assert one or more sentences to KB

- Sentence α, variable x
- Constant k (do not appear elsewhere in KB)

$$\frac{\exists x \;\; \alpha}{\mathrm{SUBST}(\{x/k\}, \alpha)}$$

Existential Instantiation

$\exists x\ Crown(x) \wedge OnHead(x, John)$

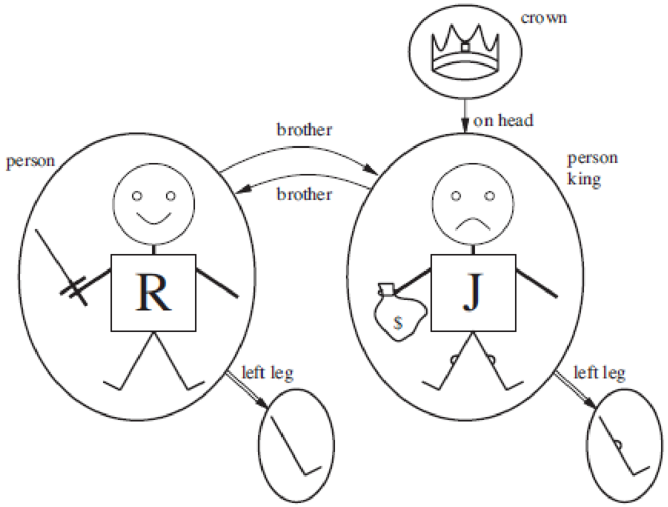
- If C_1 does not appear elsewhere in KB

  $Crown(C_1) \wedge OnHead(C_1, John)$

  $\alpha = Crown(x) \wedge OnHead(x, John)$

  $k = C_1$

$$\frac{\exists x \quad \alpha}{\text{SUBST}(\{x/k\}, \alpha)}$$



Reduction to Propositional Inference

Remove quantifier

- Existential quantifier
  - Find / create an *unseen* ground term from domain
  - Replace the variable
  - Add this new sentence to KB
- Universal quantifier
  - Find all ground terms from KB
  - Replace the variable
  - Add the set of new sentences to KB

Reduction to Propositional Inference

$$\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John) .$$

First sentence

- Apply UI with vocabulary of KB
- Two objects – {x / John}, {x / Richard}

  King(John) ∧ Greedy(John) ⇒ Evil(John)

  King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)

View all facts as propositional variables

- Use inference to induce *Evil(John)*

Propositionalization

Apply to *quantified* sentence in KB

Obtain a KB
- Consist of propositional sentences only
- Without quantifiers and variables

Very inefficient in inference
- Generates many other useless sentences
- E.g. the second asserted sentence is useless

  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

Unification

$$\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John) .$$

Only produce necessary sentences

King(John) ∧ Greedy(John) and

∀x King(x) ∧ Greedy(x) ⇒ Evil(x)

Substitution θ

- Apply on two sentences to make them look the same
- SUBST(θ, King(John) ∧ Greedy(John))
- SUBST(θ, King(x) ∧ Greedy(x))
- θ = {x / John} is a unification

Unification

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John) \, .$$

θ = {x / John}

- King(x) ∧ Greedy(x) ⇒ Evil(x)
- King(John) ∧ Greedy(John) ⇒ Evil(John)

With M.P. using King(John) and Greedy(John)

- Conclude Evil(John)

Inefficient, requires several steps

- Weakness of M.P.

Generalized Modus Ponens (GMP)

Rule capturing previous steps

Generalization of Modus Ponens

- Lifted version of M.P.

For atomic sentences p_i', p_i, and q,

- If there is a substitution θ
  - SUBST(θ, p_i)= SUBST(θ, p_i'), for all i:

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

$$\frac{\text{King(John)} \quad \text{Greedy(John)} \qquad King(x) \wedge Greedy(x) \Rightarrow Evil(x)}{\text{SUBST}(\theta = \{x / \text{John}\}, \text{Evil(x)})}$$

How to find θ?

Unification

- Take two atomic sentences *p* and *q*
- Return a substitution θ
  - Make *p* and *q* **look the same**
  - Returns **fail** if no such substitution
- UNIFY*(p, q)*=θ
  - SUBST*(*θ*, p)* = SUBST*(*θ*, q)*
  - θ is unifier of the two sentences

$$\text{UNIFY}(Knows(John, x),\ Knows(John, Jane)) = \{x/Jane\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Bill)) = \{x/Bill, y/John\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(x, Elizabeth)) = fail\ .$$

Standardizing apart

$$\text{UNIFY}(Knows(John, x), Knows(x, Elizabeth)) = fail .$$

UNIFY fail in finding θ

- Two sentences use the same variable name x
- Standardizing apart
  - Assign them with different names internally
  - In procedure of UNIFY

$$\text{UNIFY}(Knows(John, x), Knows(z_{17}, Elizabeth)) = \{x/Elizabeth, z_{17}/John\}$$

Most Generalized Unifier

May be many unifiers θ for two sentences

- The one with less constraints
- e.g. UNIFY(Knows(John, x), Knows(y, z))
- θ = { y/John, x/John, z/John }
- θ = { y/John, z/x } – the best
  - z and x are not yet found / instantiated
  - Provides greatest flexibility
  - Fewest constraints

Forward Chaining & Backward Chaining

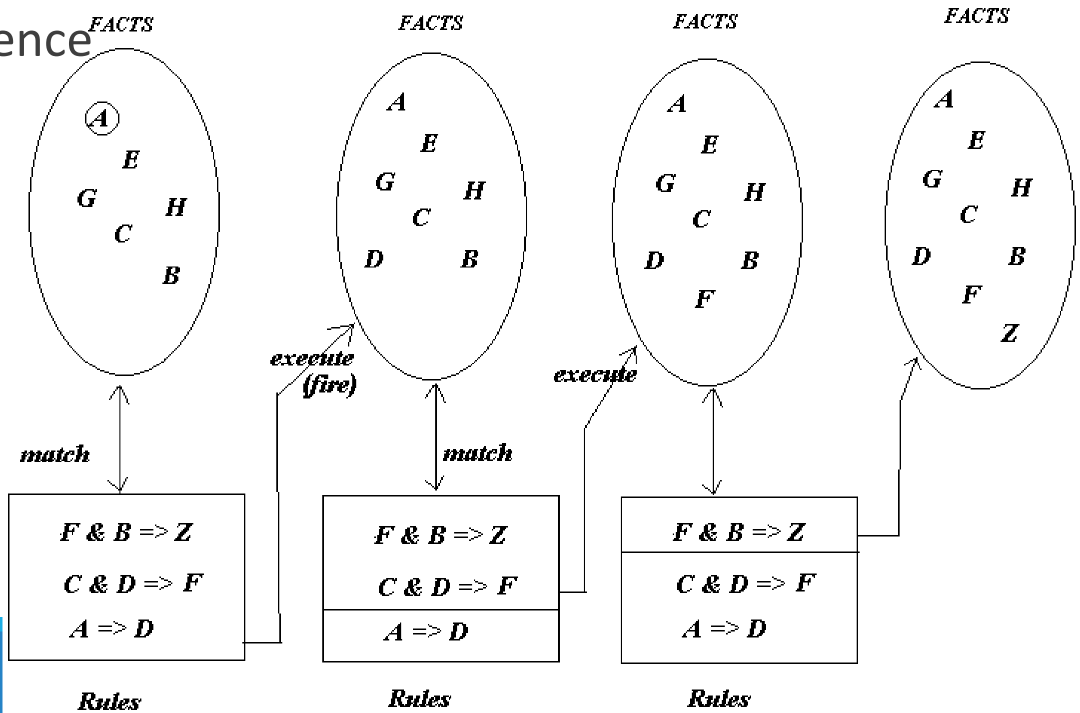
Forward Chaining

Start with sentences in KB

- Generate new conclusions
  - Make more inferences

Usually applied

- Want to generate consequence
- From new fact added to KB



Applying Forward Chaining & Backward Chaining

Convert FOL sentences into normal form

First-order Definite Clauses (Prolog)

- Can contain variables (but P.L. no variables)
- Atomic or implication
- Implication
  - Antecedent is a conjunction of **positive** literals
  - Consequent is a single **positive** literal

King(x) ∧ Greedy(x) ⇒ Evil(x) .

King(John) .

Greedy(y) .

Restriction on single positive literal

- Cannot convert every KB into a set of definite clauses, but many can

Example

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 West is a Criminal

Steps

- Translate these facts as first-order definite clauses
- Forward chaining to do inference

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

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x) . \tag{9.3}$$

"Nono ... has some missiles." The sentence $\exists\, x\ Owns(Nono, x) \land Missile(x)$ is transformed into two definite clauses by Existential Instantiation, introducing a new constant M_1:

$$Owns(Nono, M_1) \tag{9.4}$$

$$Missile(M_1) \tag{9.5}$$

"All of its missiles were sold to it by Colonel West":

$$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono) . \tag{9.6}$$

We will also need to know that missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x) \tag{9.7}$$

and we must know that an enemy of America counts as "hostile":

$$Enemy(x, America) \Rightarrow Hostile(x) . \tag{9.8}$$

"West, who is American ...":

$$American(West) . \tag{9.9}$$

"The country Nono, an enemy of America ...":

$$Enemy(Nono, America) . \tag{9.10}$$

Forward Chaining

- On the first iteration, rule (9.3) has unsatisfied premises.
  Rule (9.6) is satisfied with $\{x/M_1\}$, and $Sells(West, M_1, Nono)$ is added.
  Rule (9.7) is satisfied with $\{x/M_1\}$, and $Weapon(M_1)$ is added.
  Rule (9.8) is satisfied with $\{x/Nono\}$, and $Hostile(Nono)$ is added.

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

$$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x) . \tag{9.3}$$

"All of its missiles were sold to it by Colonel West":

$$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono) . \tag{9.6}$$

We will also need to know that missiles are weapons:

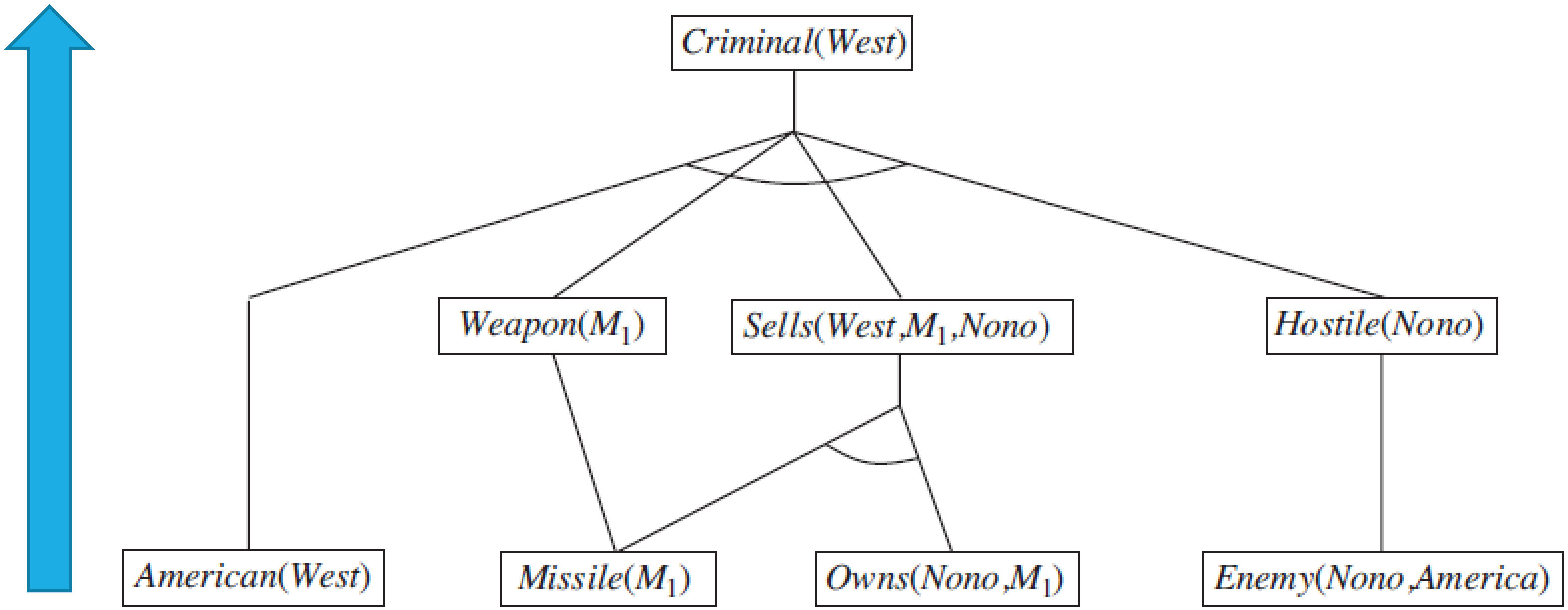
$$Missile(x) \Rightarrow Weapon(x) \tag{9.7}$$

and we must know that an enemy of America counts as "hostile":

$$Enemy(x, America) \Rightarrow Hostile(x) . \tag{9.8}$$

- On the second iteration, rule (9.3) is satisfied with $\{x/West, y/M_1, z/Nono\}$, and $Criminal(West)$ is added.

Proof Tree



No new inferences can be made using current KB

- Fixed point of inference process

Backward Chaining

Start with something want to prove

- Goal / query

Look for the implication sentences

- Would conclude the goal

Attempt to establish their premises

Normally used

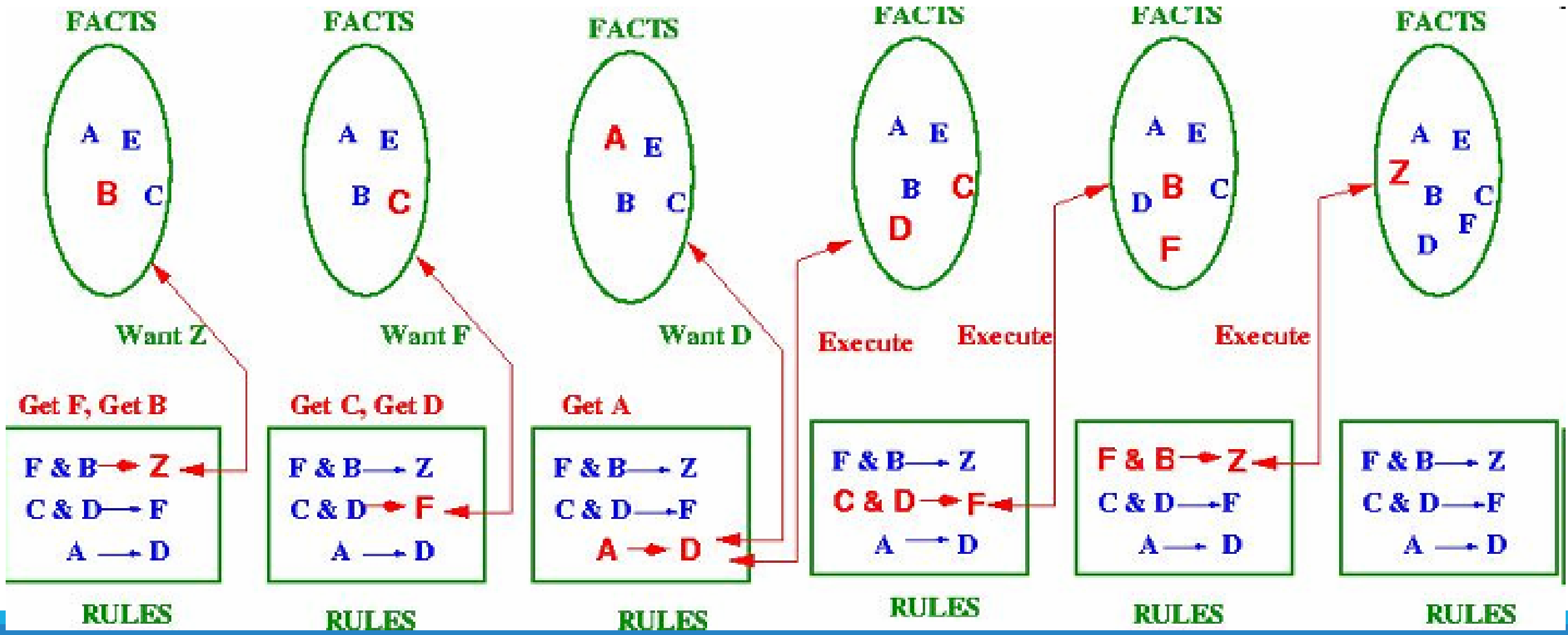
- When there is a goal to prove or query
- Prolog

Backward Chaining

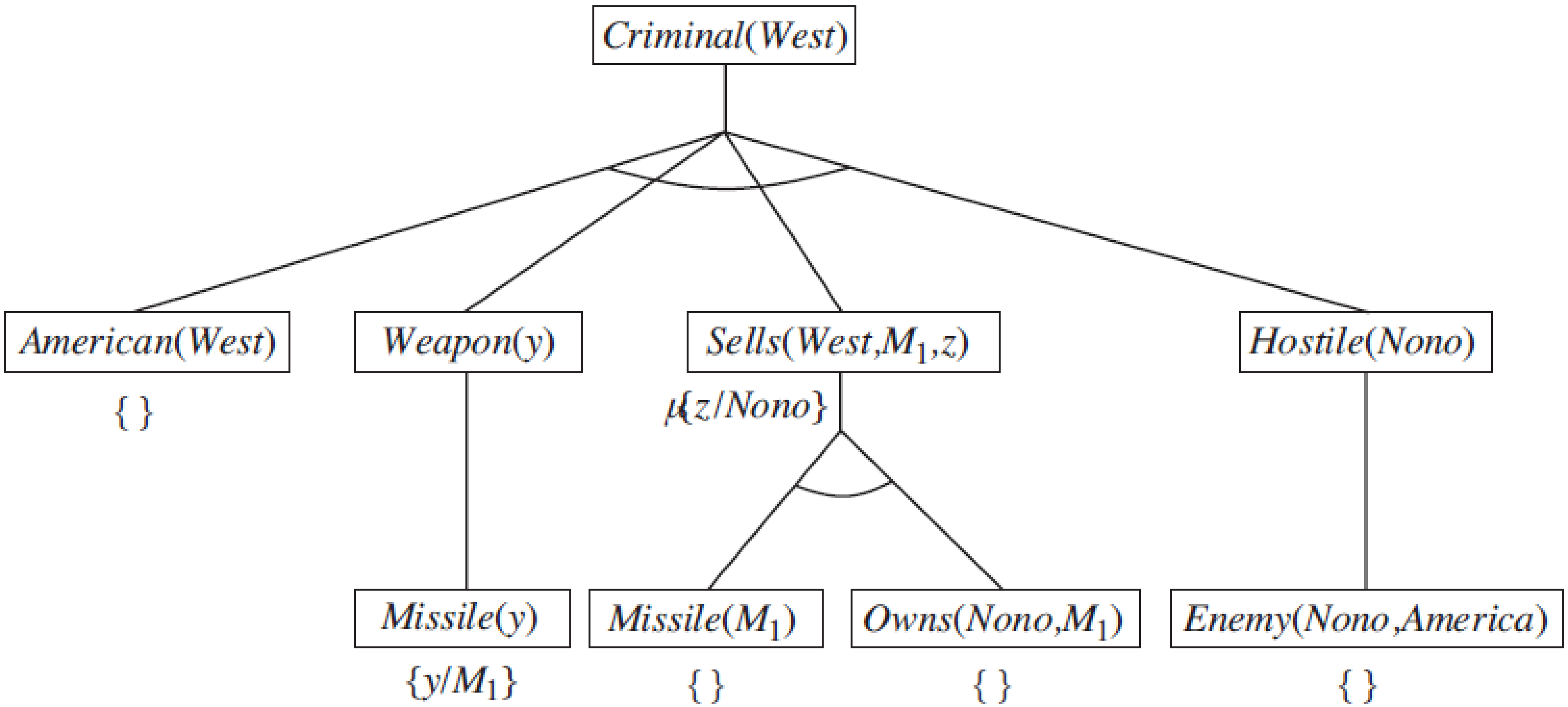
Look for the implication sentences

- Would conclude the goal

Attempt to establish their premises



Proof Tree



Criminal(West)
American(West)
{ }
Weapon(y)
Sells(West,M1,z)
{z/Nono}
Hostile(Nono)
Missile(y)
{y/M1}
Missile(M1)
{ }
Owns(Nono,M1)
{ }
Enemy(Nono,America)
{ }

Backward Chaining

Uses composition of substitutions

- $\mathrm{SUBST}(\mathrm{COMPOSE}(\theta_1, \theta_2), p)$

$= \mathrm{SUBST}(\theta_2, \mathrm{SUBST}(\theta_1, p))$

Different goals

- Different unifications are found
- Combine them

Resolution

Resolution

Modus Ponens rule

- Only can derive atomic conclusion
- $\{A, A \Rightarrow B\} \vdash B$

Natural to derive new implication

- $\{A \Rightarrow B, B \Rightarrow C\} \vdash A \Rightarrow C$, transitivity
- More powerful tool: **resolution rule**

Conjunctive Normal Form

CNF for FOL

- A conjunction (AND) of clauses
  - Each is a disjunction (OR) of literals
  - *Literals can contain variables*
- E.g.

$$\forall x \;\; American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \;\Rightarrow\; Criminal(x)$$

becomes, in CNF,

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

Conversion to CNF

$$\forall x \ [\forall y \ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y \ \ Loves(y,x)] .$$

◇ **Eliminate implications**:

$$\forall x \ [\neg\forall y \ \neg Animal(y) \lor Loves(x,y)] \lor [\exists y \ \ Loves(y,x)] .$$

◇ **Move $\neg$ inwards**: In addition to the usual rules for negated connectives, we need rules for negated quantifiers. Thus, we have

$$\begin{array}{lll} \neg\forall x \ \ p & \text{becomes} & \exists x \ \ \neg p \\ \neg\exists x \ \ p & \text{becomes} & \forall x \ \ \neg p . \end{array}$$

Our sentence goes through the following transformations:

$$\begin{array}{l} \forall x \ [\exists y \ \ \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y \ \ Loves(y,x)] . \\ \forall x \ [\exists y \ \ \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y \ \ Loves(y,x)] . \\ \forall x \ [\exists y \ \ Animal(y) \land \neg Loves(x,y)] \lor [\exists y \ \ Loves(y,x)] . \end{array}$$

◇ **Standardize variables**: For sentences like $(\forall x \ \ P(x)) \lor (\exists x \ \ Q(x))$

$$\forall x \ [\exists y \ \ Animal(y) \land \neg Loves(x,y)] \lor [\exists z \ \ Loves(z,x)]$$

Conversion to CNF

Skolemize

- Process of removing $\exists$
- Translate $\exists$x P(x) into P(A), A is a new constant

$$\forall x \;\; [Animal(A) \land \neg Loves(x, A)] \lor Loves(B, x)$$

  - Completely wrong
  - Since A is a certain animal (a constant)
  - Use a function to represent any animal
    - Skolem function

$$\forall x \;\; [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

Conversion to CNF

Universal quantifiers

- Drop it
- Assume all variables to be universally quantified now

All the steps can be automated

◇ **Drop universal quantifiers:** At this point, all remaining variables must be universally quantified. Moreover, the sentence is equivalent to one in which all the universal quantifiers have been moved to the left. We can therefore drop the universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) .$$

◇ **Distribute** $\land$ **over** $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)] .$$

Resolution Inference Rule

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\textsc{Subst}(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$. For example, we can resolve the two clauses

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

by eliminating the complementary literals $Loves(G(x), x)$ and $\neg Loves(u, v)$, with unifier $\theta = \{u/G(x), v/x\}$, to produce the **resolvent** clause

$$[Animal(F(x)) \vee \neg Kills(G(x), x)] .$$

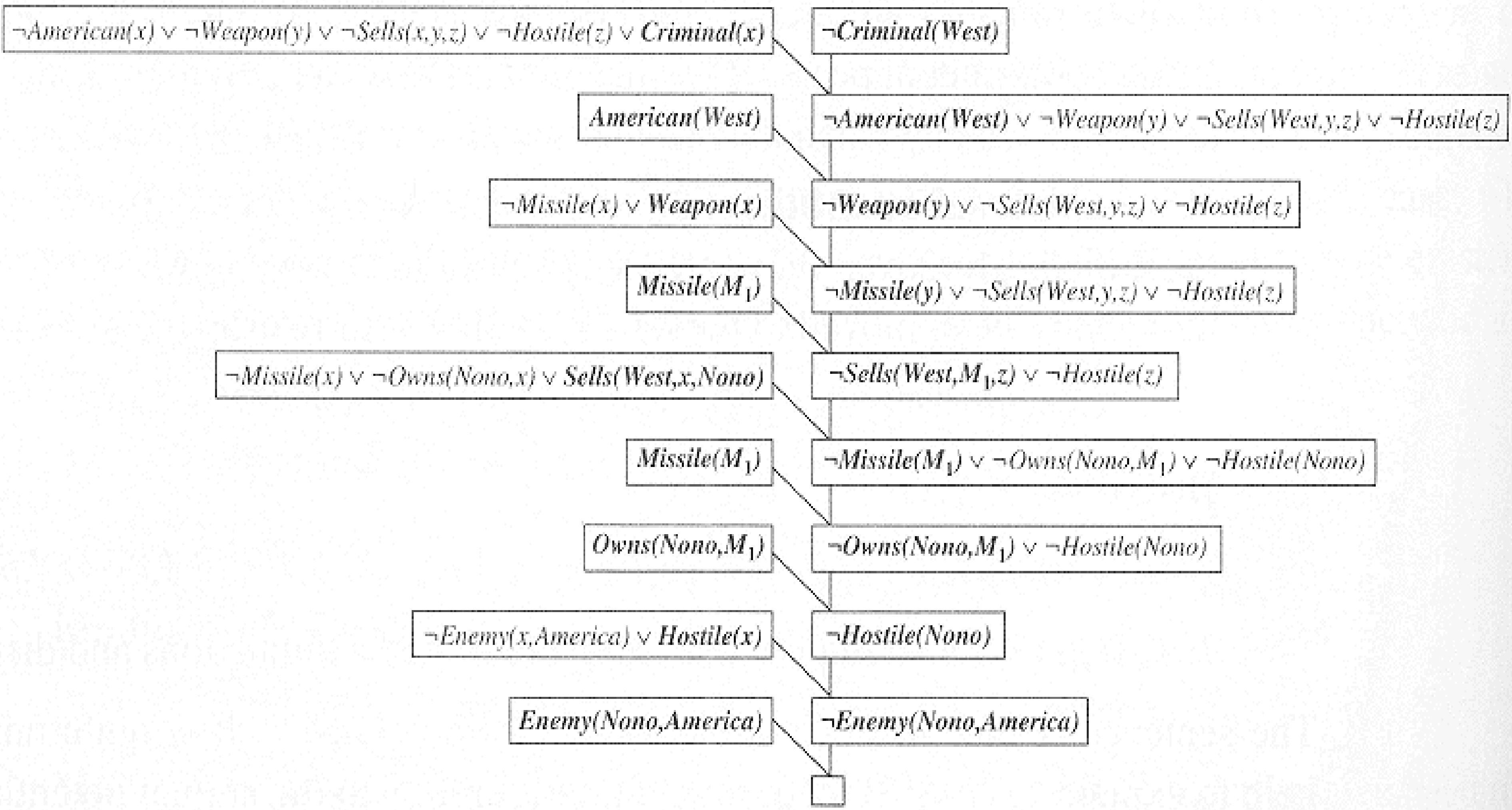
Example Proof

Resolution proves that KB |= α

- Prove KB ∧ ¬α unsatisfiable, i.e. empty clause
- First convert the sentences into CNF

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee Criminal(x)$$
$$\neg Missile(x) \vee \neg Owns(Nono,x) \vee Sells(West,x,Nono) .$$
$$\neg Enemy(x, America) \vee Hostile(x) .$$
$$\neg Missile(x) \vee Weapon(x) .$$
$$Owns(Nono, M_1) . \qquad Missile(M_1) .$$
$$American(West) . \qquad Enemy(Nono, America) .$$

- Empty clause
  - Conclude the negated goal ¬ Criminal(West)
  - i.e. Criminal(West)


¬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)

Example Proof

Another example involves

- Skolemization, non-definite clause
- Make inference more complex

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

First, we express the original sentences, some background knowledge, and the negated goal G in first-order logic:

A. $\forall x \; [\forall y \; Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y \; Loves(y, x)]$

B. $\forall x \; [\exists y \; Animal(y) \wedge Kills(x, y)] \Rightarrow [\forall z \; \neg Loves(z, x)]$

C. $\forall x \; Animal(x) \Rightarrow Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x \; Cat(x) \Rightarrow Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

First, we express the original sentences, some background knowledge, and the negated goal G in first-order logic:

A. $\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ \ Loves(y, x)]$

B. $\forall x\ [\exists y\ \ Animal(y) \wedge Kills(x, y)] \Rightarrow [\forall z\ \ \neg Loves(z, x)]$

C. $\forall x\ \ Animal(x) \Rightarrow Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x\ \ Cat(x) \Rightarrow Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

Now we apply the conversion procedure to convert each sentence to CNF:

A1. $Animal(F(x)) \vee Loves(G(x), x)$

A2. $\neg Loves(x, F(x)) \vee Loves(G(x), x)$

B. $\neg Animal(y) \vee \neg Kills(x, y) \vee \neg Loves(z, x)$

C. $\neg Animal(x) \vee Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

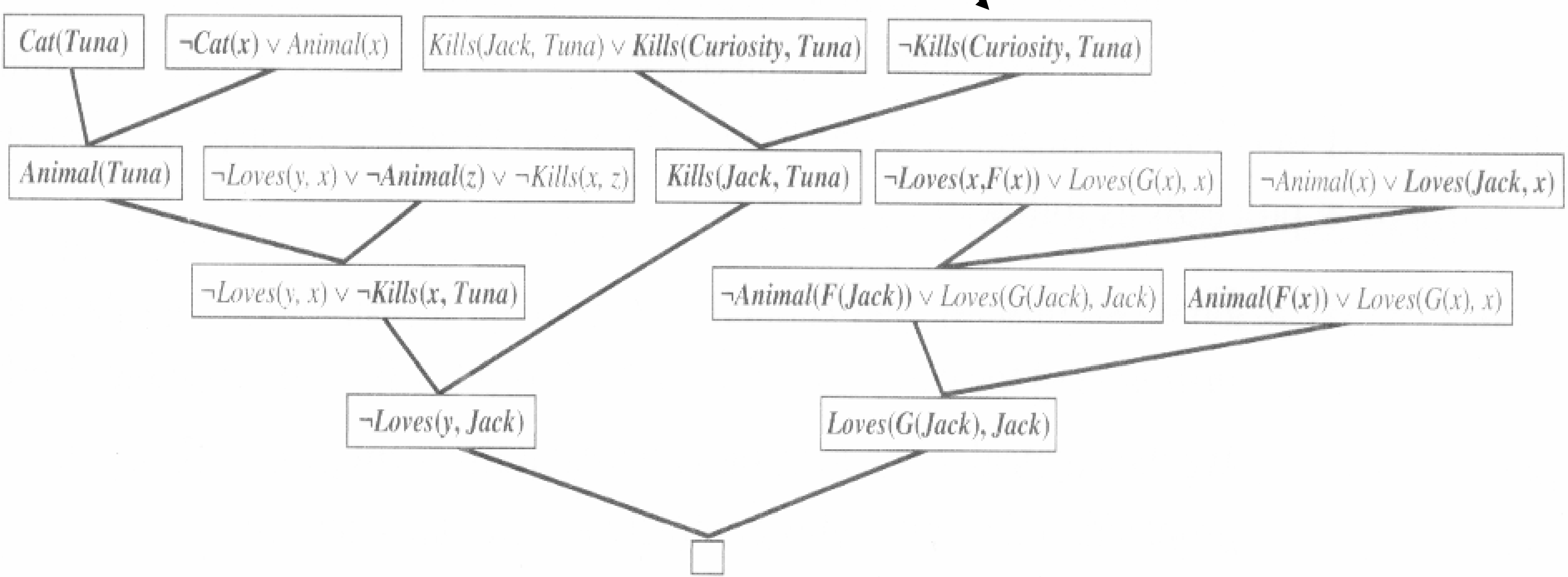
F. $\neg Cat(x) \vee Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

Query: Did Curiosity kill the cat? α

Assume: Curiosity didn't kill the cat $\neg\alpha$

$KB \wedge \neg\alpha$



Empty clause, so assumption is false, α is true

Resolution Strategies

Resolution
- Effective but very inefficient

Like forward chaining
- Reasoning by randomly tried

Four general guidelines
- Unit Preference
- Set of Support
- Input Resolution
- Subsumption

Unit Preference

Resolution on two sentences

One must be a unit clause

- i.e. an atomic sentence
- king(John), missle(M1), …

Idea

- Produce a shorter sentence
  - E.g. $P \vee Q \vee R$ and $\neg P$
  - Produce $Q \vee R$
- Reduce complexity of clauses

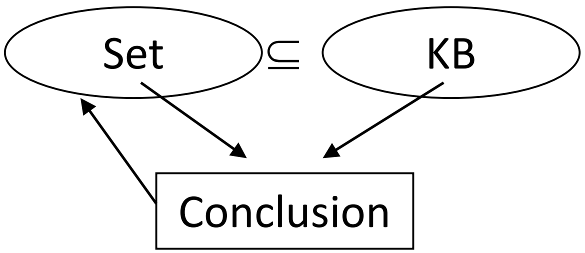
Set of Support

Identify a subset of sentences from KB

Resolution combines a sentence
- From the subset
- From KB

Conclusion of the resolution
- Add to the subset
- Continue the resolution process

Identify the subset
- The negated query
  - Query to be proved, assume negative
  - Prove by contradiction
- Advantage: goal-directed



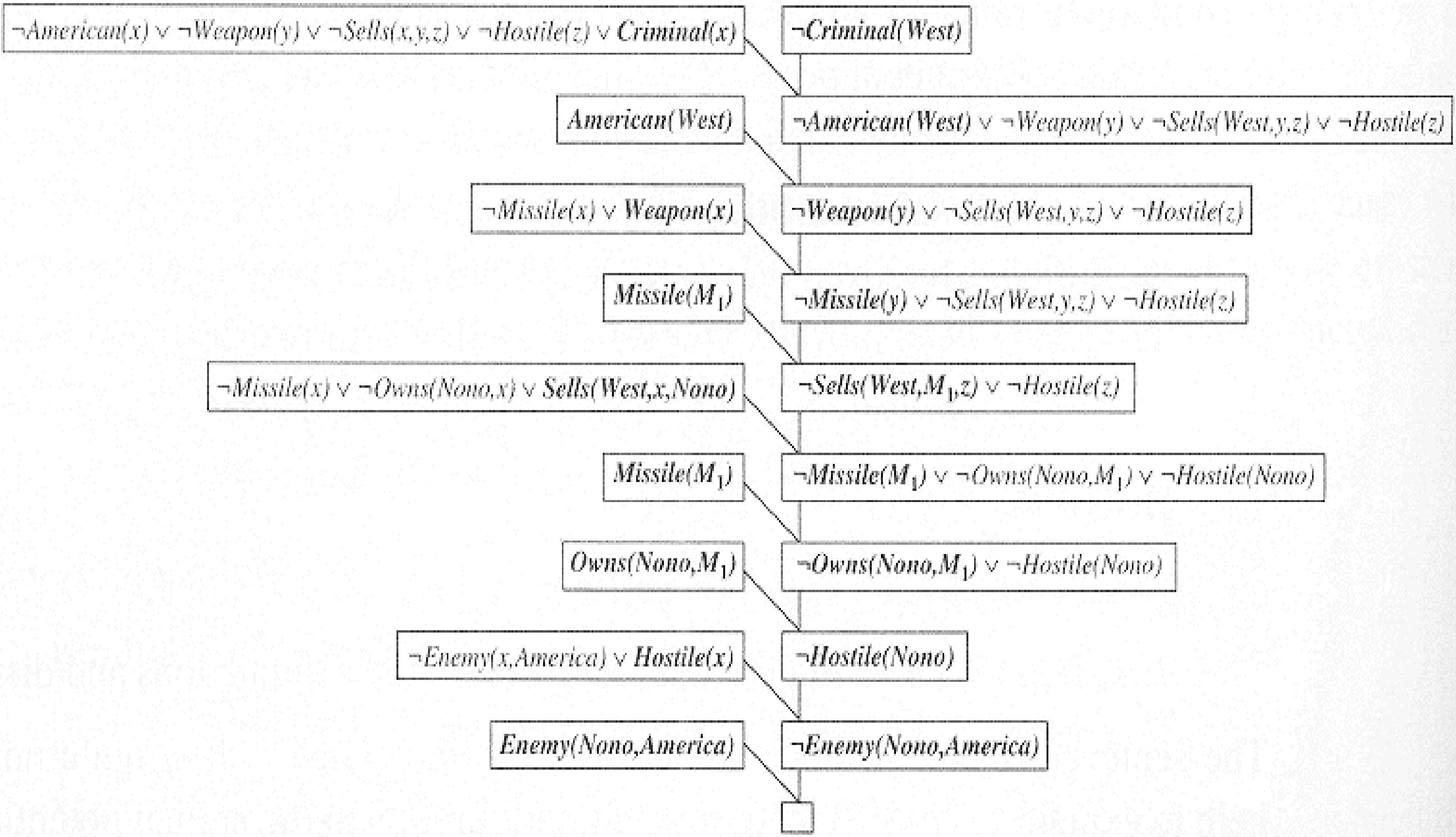
Input Resolution

Resolution combines a sentence

- From the ***input*** sentences
  - The query
  - KB
- Some other sentence
  - Including conclusion from resolution procedure

Idea

- Make conclusion related to the query or KB
- Not to use two newly concluded sentences



Each resolution

- At least one sentence from query or KB

Subsumption (Inclusion)

Eliminates all sentences

- Subsumed by an existing sentence in KB
- i.e. Use a more general sentence instead of many specific rules

For example

- *P(x)* is in KB where *x* is a variable
- Do not need to store specific instances
  - *P(A)*, *P(B)*, *P(C)* …

Keep KB small