# Knowledge Representation and Reasoning

## *Lecture 7: FOL*

Dr Khaled Alrfou

Chapters: 8, 9 and 10

Tafila Technical University

E-mail: Khaled@stu.ttu.edu.jo

# LOGIC

# Knowledge representation

- **Knowledge representation and reasoning** is the part of Artificial intelligence
- It is responsible for representing information about the real world
- It is also a way which describes how we can represent knowledge in artificial intelligence.

# Knowledge Representation - Example

1. If it does not rain, then I will be at the university today.
2. Today, I will either be at the university or at home, but not both.
3. I am at home today.

# Knowledge Representation - Example

1. If it does not rain, then I will be at the university today.
2. Today, I will either be at the university or at home, but not both.
3. I am at home today.
4. **I am not the university today**

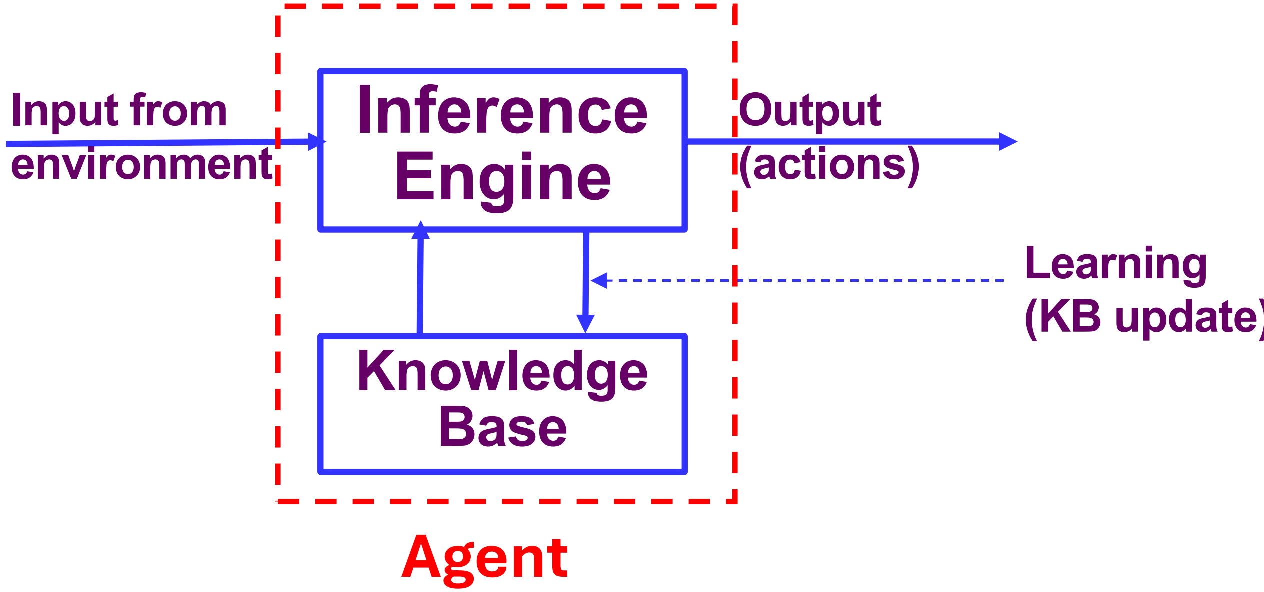
# Knowledge Representation - Example

1. If it does not rain, then I will be at the university today.
2. Today, I will either be at the university or at home, but not both.
3. I am at home today.
4. **I am not the university today**
5. **It does rain today**

# A Knowledge-Based Agent

- A knowledge-based agent consists of a **knowledge base** (KB) and an **inference engine** (IE)

- A knowledge base is **a set of sentences.**
  - Here "sentence" is used as a technical term. It is related but not identical to the sentences of English and other natural languages.
- Each **sentence** is expressed in a language called a **knowledge representation language** and represents some assertion about the world.
- Sentences that were not derived from others are called **axioms**



# Propositional Logic

- Propositional logic assumes the world contains only statements with a truth value.
- Statements can be combined with binary connectives (AND, OR) or negated. For example:
  - A V B V ~C
- Proofs can use truth tables, or by rewriting one expression as another via natural deduction (e.g. DeMorgan’s laws)
- But, PL is not expressive enough for many problems.

# Logics in general

| Language | Ontological Commitment | Epistemological Commitment |
| --- | --- | --- |
| Propositional logic | facts | true/false/unknown |
| First-order logic | facts, objects, relations | true/false/unknown |
| Temporal logic | facts, objects, relations, times | true/false/unknown |
| Probability theory | facts | degree of belief |
| Fuzzy logic | facts + degree of truth | known interval value |

# Propositional logic

$((A \land C) \lor (A \Rightarrow C)) \Rightarrow (C \lor B)$

| A | B | C | $((A \land C) \lor (A \Rightarrow C)) \Rightarrow (C \lor B)$ |
|---|---|---|---|
| T | T | T | T |
| T | T | F | T |
| T | F | T | T |
| T | F | F | T |
| F | T | T | T |
| F | T | F | T |
| F | F | T | T |
| F | F | F | F |

*satisfiable, but not valid*

$(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

| A | B | C | $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ |
|---|---|---|---|
| T | T | T | T |
| T | T | F | T |
| T | F | T | T |
| T | F | F | T |
| F | T | T | T |
| F | T | F | T |
| F | F | T | T |
| F | F | F | T |

*valid*

$(A \lor B) \land (A \lor \neg B) \land (\neg A \lor B) \land (\neg A \lor \neg B)$

| A | B | $(A \lor B) \land (A \lor \neg B) \land (\neg A \lor B) \land (\neg A \lor \neg B)$ |
|---|---|---|
| T | T | F |
| T | F | F |
| F | T | F |
| F | F | F |

*unsatisfiable*

# Inference rules for propositional logic

Modus Ponens $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$

And-Elimination (AE): $\dfrac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$

And-Introduction (AI): $\dfrac{\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$

Or-Introduction (OI): $\dfrac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$

Double-Negation Elimination (DNE): $\dfrac{\neg\neg\alpha}{\alpha}$

Unit Resolution (UR): $\dfrac{\alpha \vee \beta,\ \neg\beta}{\alpha}$

Resolution (R): $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

deMorgan's Law (DML): $\dfrac{\neg(\neg\alpha \vee \beta)}{\alpha \wedge \neg\beta}$

Given the following knowledge base:

1. P
2. P $\Rightarrow$ R
3. R $\Rightarrow$ $\neg$W
4. S $\vee$ R
5. (P $\wedge$ R) $\Rightarrow$ (S $\vee$ W)

Prove **S** using natural deduction with these rules.

6. R (MP: 1,2)
7. $\neg$W (MP: 3,6)
8. P $\wedge$ R (AI: 1,6)
9. S $\vee$ W (MP: 5,8)
10. S (UR: 7,9)

# First-Order Logic

- First-order logic assumes the world contains
  - Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries
  - Relations: red, round, bogus, prime, multistoried, brother of, bigger than, inside, part of, has color, occurred after, owns, comes between,
  - Functions: father of, best friend, third inning of, one more than, beginning of.

# Syntax

- The syntax of a logic tells you the well-formed sequences of symbols

- FOL has 2 types of symbols:
  - **Logical symbols** have a fixed meaning in the language. There are 3 types:
    - Punctuation (such as parentheses)
    - Connectives and quantifiers
    - Variables (x, y, z, x1, x2, ...)

  - **Non-logical symbols** are those whose meanings depend on the application (there are an infinite supply of these)
    - Function symbols (bestFriend, a, b, c, d,..., h)
    - Predicate symbols (OlderThan, P, Q, R, P1,...)

# *Terms*

- A term is used to denote an object in the world
  - **Variables: `x`, `y`, `z`, …**
  - **Function($term_1$, …, $term_n$):**
    - **Constants: `(functions of arity 0): Max` …**
    - *e.g.* **`sqrt(9), distance(ALB,x)`**
    - Use functions to express names or to refer to an unnamed object: e.g. **`leftLegOf(John)`**

- A ground term is a term with no variables

# *Formulas*

- An atomic sentence is the smallest expression to which a truth value can be assigned
  - **Predicate($term_1$, …, $term_n$):**
    - *e.g.* **`teacher(Khaled,You)`**, **`lte(sqrt(2),sqrt(7))`**
    - Maps one or more objects to a truth value
    - Represents a user defined relation
  - **$Term_1$ = $Term_2$:**
    - *e.g.* **`height(Jim) = 66`**, **`1 = 2`**
    - Represents when two terms refer to the same object

# *Formulas (2)*

- A sentence represents a fact in the world that is assigned a truth value; It is:
  - Atomic sentence
  - Complex sentence using connectives:
    - *e.g.* **spouse(George,Laura) ↔ spouse(Laura,George)**
  - Complex sentence using quantified variables: ∀ ∃ sentence
    - *e.g.* $\forall$ **x** $\forall$ **y (spouse(x,y) ↔ spouse(y,x))**

# Syntax: The Connectives

- If $S$ is a sentence $\neg S$ is a sentence
- If $S_1$ and $S_2$ are sentences, $S_1 \wedge S_2$ is a sentence
- If $S_1$ and $S_2$ are sentences, $S_1 \vee S_2$ is a sentence
- If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence
- If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence

# *Non-logical symbols*

- The vocabulary will define the domain-dependent predicates and functions.
- We can name individual people, places, etc.
    - Such as: John, Jane, ALB, Aurora
- We will also be able to define for them:
    - Basic types: e.g. Person, Man, Place, Company
    - Attributes: e.g. Rich, Beautiful, etc
    - Relationships: Married, LivesIn, WorksFor, etc
    - Functions: WifeOf,  FounderOf, etc

# *Expressing Basic Facts*

- The basic facts of a problem can usually be expressed as atomic sentences.
  - Type facts
    - Person(John), Person(Jane), Company(Aurora)
  - Property facts
    - Married(John,Jane), WorksFor(John,Aurora)
  - Equality facts
    - Jane = WifeOf(John)

# *Complex Facts*

- Complex facts will require using connectives.
  - These would include universals:
    - `∀ x ManagerOf(x,Aurora) ⇒ Rich(x)`
    - `∀ x Rich(x) & WorksFor(x,Aurora) ⇒ Lives(x,RiverHills)`
  - And uncertain facts
    - `LivesIn(John, Milwaukee)V LivesIn(John, RiverHills)`
  - And closure axioms
    - `∀ x Person(x) ⇒ x=John V x=Jane`
    - `Jane ≠ John`

# *Terminological Facts*

- These are general relationships among predicates, such as:
  - disjointness $\forall x [Man(x) \Rightarrow \neg Woman(x)]$
  - subtype $\forall x [Surgeon(x) \Rightarrow Doctor(x)]$
  - exhaustive $\forall x [Adult(x) \Rightarrow Man(x) \vee Woman(x)$
  - symmetry $\forall x \forall y [MarriedTo(x,y) \Rightarrow MarriedTo(y,x)]$
  - inverses $\forall x \forall y [ChildOf(x,y) \Rightarrow ParentOf(y,x)]$
  - type restrictions $\forall x \forall y [MarriedTo(x,y) \Rightarrow$
    $Person(x) \wedge Person(y) \wedge x \neq y ]$
- They are usually universally quantified conditionals or biconditionals

# *Using Functions and Equality*

- "Mike and Mary are the same age."
  - Are functional relations specified?
  - Are equalities specified?
  - **Answer:** $\mathbf{age(Mike) = age(Mary)}$
- "There are exactly two shoes."
  - Are quantities specified?
  - Are equalities implied?
  - **Answer:** $\exists x\ \exists y\ shoe(x) \land shoe(y) \land \neg(x=y) \land \forall z\ (shoe(z) \Rightarrow (x=z) \lor (y=z))$

# Semantics of Sentences

- The domain will allow us to compute the truth value for each proposition:

  *e.g.* $\mathbf{P}_1$ = true, $\mathbf{P}_2$ = false

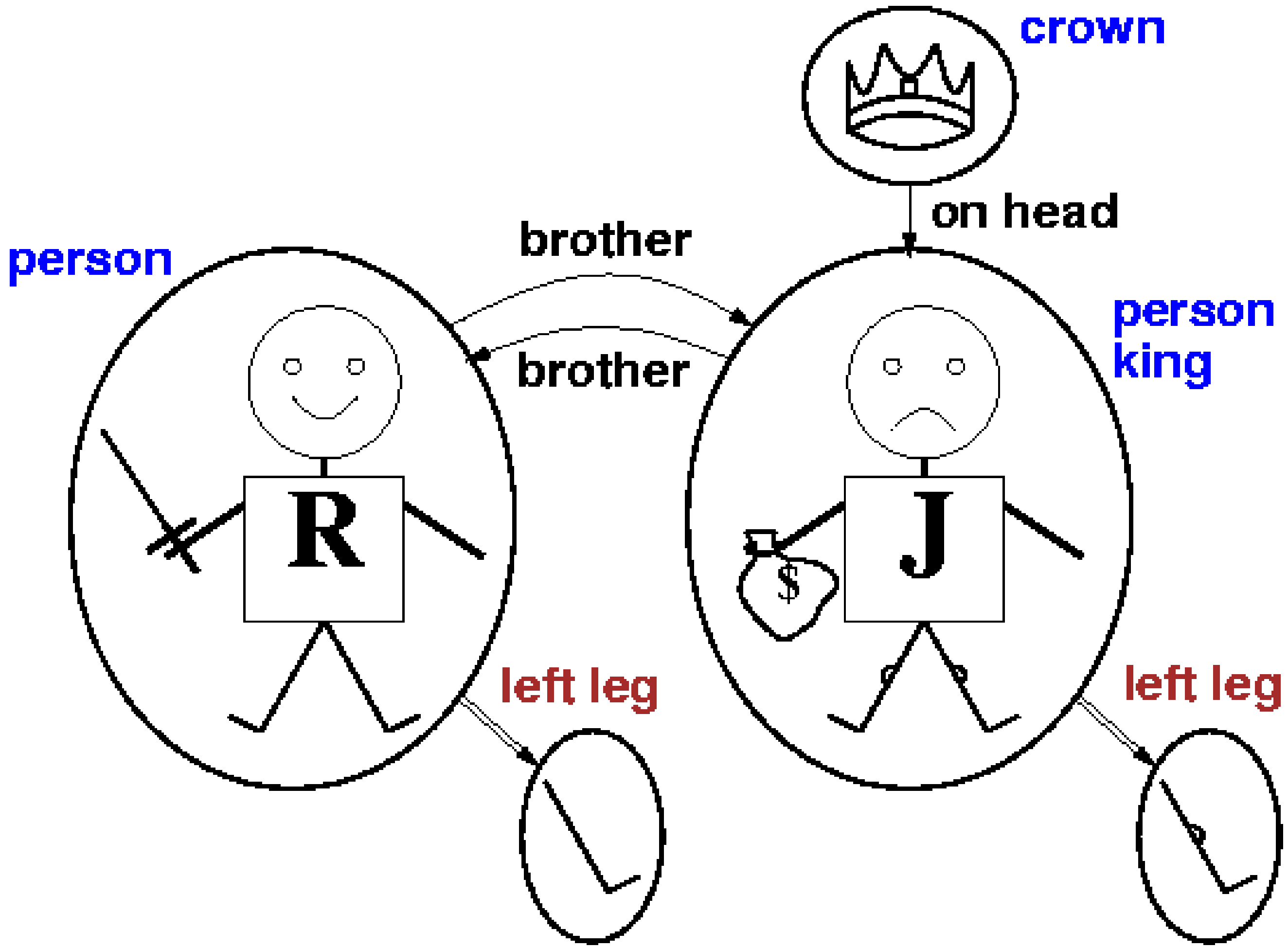
- We also use models to interpret expressions of equality, ex f(a) = f(b) iff I(f(a)) and I(f(b)) are the same in the model.

- To define connectives, we have rules for evaluating truth with respect to some model $\mathbf{m}$:

| | | | | | |
|---|---|---|---|---|---|
| $\neg S$ | is true iff $S$ | is false | | | |
| $S_1 \wedge S_2$ | is true iff $S_1$ | is true | and | $S_2$ | is true |
| $S_1 \vee S_2$ | is true iff $S_1$ | is true | or | $S_2$ | is true |
| $S_1 \Rightarrow S_2$ | is true iff $S_1$ | is false | or | $S_2$ | is true |
| | is false iff $S_1$ | is true | and | $S_2$ | is false |
| $S_1 \Leftrightarrow S_2$ | is true iff $S_1 \Rightarrow S_2$ | is true | and | $S_2 \Rightarrow S_1$ | is true |

- Operator Precedence: **(highest)** $\neg \wedge \vee \Rightarrow \Leftrightarrow$ **(lowest)**

# An Example


crown
on head
brother
person
brother
person
king
R
J
$
left leg
left leg

# Assigning Truth

*The atomic sentence $predicate(term_1, \ldots, term_n)$ is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by the predicate*

- What is the truth value for `holding(J,cash1)`?
  - Domain:
    - **Objects: R, J, sword1, cash1, …**
    - **Relation: holding is {<R,sword1>,<J,cash1>}**
  - Since **<J,cash1>** is in the holding relation, `holding(J,cash1)` is true.

# *Universal Quantifiers*

The universal quantifier: $\forall$

✱*Sentence holds true* ***for all*** *values of* x *in the domain of variable* x

- This quantifier is often used with the connective $\Rightarrow$ (the same as $\supset$) to form if-then rules
  - "All humans are mammals" in FOL becomes:
    $\forall$**`x human(x)`** $\Rightarrow$ **`mammal(x)`**
  - Means " if *x* is a human then *x* is a mammal"

# *Universal Quantifiers(2)*

$\forall x\ \mathtt{human}(x) \Rightarrow \mathtt{mammal}(x)$

- Equivalent to the conjunction of all the instantiations of variable *x*:

  $(\mathtt{human}(\mathtt{Rodgers}) \Rightarrow \mathtt{mammal}(\mathtt{Rodgers}))\ \&$
  $(\mathtt{human}(\mathtt{Cutler}) \Rightarrow \mathtt{mammal}(\mathtt{Cutler}))\ \&$
  $(\mathtt{human}(\ldots) \Rightarrow \mathtt{mammal}(\ldots))\ \&\ \ldots$

# *Existential Quantifiers*

The existential quantifier: ∃

✱*Sentence holds true **for some** value of x in the domain of variable x*

- Main connective typically ∧ **(same as "&")**
  - "Some humans are rich" in FOL becomes:

    **∃x human(x) ∧ rich(x)**
  - Means *x* is some human and *x* is rich.

# *Existential Quantifiers (2)*

$\exists$`x human(x) ∧ rich(x)`

- Equivalent to the disjunction of all the instantiations of variable *x*:

```
(human(Rodgers) ∧ rich(Rodgers)) ∨
(human(Cutler) ∧ rich(Cutler)) ∨
(human(my_cat) ∧ rich(my_cat)) ∨ …
```

# *Negated Quantifiers*

- Properties of quantifiers:
  - $\forall x\ P(x)$ when negated is $\exists x\ \neg P(x)$
  - $\exists x\ P(x)$ when negated is $\forall x\ \neg P(x)$

- Why?
  - $\forall x\ \mathrm{sleeps}(x)$
    “Everybody sleeps.”
    - Negated: $\neg\ (\forall x\ \mathrm{sleeps}(x))$
  - $\exists x\ \neg \mathrm{sleeps}(x)$
    which says: “Somebody doesn’t sleep.”

## Convert the following sentences into FOL

- “Bob is a fish”
  - What is the constant?
    - Bob
  - What is the predicate?
    - Fish
  - Answer: fish(Bob)

# Thinking in Logic Sentences

## Convert the following sentences into FOL

- "Mike and Mary are grad students."
  - What is the constant?
    - Mike and Mary
  - What is the predicate?
    - Grad
  - Answer: Grad(Mike) $\wedge$ Grad(Mary)

# Thinking in Logic Sentences

## We can also do this with relations

- “America bought Alaska from Russia.”
  - What is the constant?
    - America, Alaska, and Russia
  - What is the predicate?
    - bought
  - Answer: bought(America, Alaska, Russia)

## Now Let's think about quantification

- "George likes everything."
  - What is the constant?
    - George
  - How are they variables quantified?
    - All/ universal
  - What is the predicate?
    - likes
  - Answer: **$\forall$x likes (George, x)**

*i.e.* `likes(George, IceCream) ∧ likes(George, Laura) ∧ likes(George, Armadillos) ∧ …`

# Thinking in Logic Sentences

## Now Let's think about quantification

- "George likes something."
  - What is the constant?
    - George
  - How are they variables quantified?
    - Existential
  - **Answer:** `∃x likes(George, x)`
  - *i.e.* `likes(George, IceCream) V likes(George, Laura) V likes(George, Armadillos) V …`

# Thinking in Logic Sentences

## Now Let's think about quantification

- All
  - **Things:** anything, everything, whatever
  - **Persons:** anybody, anyone, everybody, everyone, whoever
- Some (at least one)
  - **Things:** something
  - **Persons:** somebody, someone
- None
  - **Things:** nothing
  - **Persons:** nobody, no one

# Thinking in Logic Sentences

**We can also have multiple quantifiers:**

- "Somebody heard something."
  - What are the variables?
    - somebody and something
  - How are they quantified?
    - both are at least one/existential
  - **Answer:** $\exists$`x,y heard(x,y)`
- "Everybody heard everything."
- "Somebody did not hear everything."

# Thinking in Logic Sentences

**We can also have multiple quantifiers:**

- "Everybody heard something."
  - $\forall$ `x,` $\exists$ `y heard(x,y)`
- "Somebody did not hear everything"
  - **Answer:** $\exists$`x, not` $\forall$ `y heard(x,y)`
  - **Equivalently:** $\exists$`x,` $\exists$ `y not heard(x,y)`

# Thinking in Logic Sentences

**Let's allow more complex quantified relations:**

- "All stinky shoes are allowed."
  - How are ideas connected?
    - being a shoe *and* being stinky *implies* that it is allowed
  - **Answer:** $\forall x \ \mathtt{shoe(x)} \wedge \mathtt{stinky(x)} \Rightarrow \mathtt{allowed(x)}$
- "No stinky shoes are allowed."
  - **Answer:** $\neg\exists x \ \mathtt{shoe(x)} \wedge \mathtt{stinky(x)} \wedge \mathtt{allowed(x)}$
- The equivalent:
  "Stinky shoes are not allowed."
  - **Answer:** $\forall x \ \mathtt{shoe(x)} \wedge \mathtt{stinky(x)} \Rightarrow \neg\mathtt{allowed(x)}$

# Thinking in Logic Sentences

**And some more complex relations:**

- "No one sees everything."
  - What are the variables and quantifiers?
    - nothing and everything
    - not one (i.e. not existential) and all (universal)
  - **Answer:** $\neg\exists x\ \forall y\ \text{sees}(x,y)$
- Equivalently:
  "Everyone doesn't see something."
  - **Answer:** $\forall x\ \exists y\ \neg\text{sees}(x,y)$
- Which is different from "Everyone sees nothing."
  - **Answer:** $\forall x\ \neg\exists y\ \text{sees}(x,y)$

# Thinking in Logic Sentences

**And some *really* complex relations:**

- "Any good amateur can beat some professional."
  - Lets break this down:
    - $\forall x$ [ ($x$ is a good amateur) $\Rightarrow$ ($x$ can beat some professional) ]
    - ($x$ can beat some professional) *is really:*
      $\exists y$ [ ($y$ is a professional) $\wedge$ ($x$ can beat $y$) ]
    - $\forall x$ [ ($x$ is a good amateur) $\Rightarrow$
      $\exists y$ [ ($y$ is a professional) $\wedge$ ($x$ can beat $y$) ]
  - **Answer:** $\forall x\ [\{amateur(x) \wedge good(x)\} \Rightarrow \exists y\ \{professional(y) \wedge beat(x,y)\}]$
- "Some professionals can beat all amateurs."
  - **Answer:** $\exists x\ [professional(x) \wedge \forall y\ \{amateur(y) \Rightarrow beat(x,y)\}]$

# Thinking in Logic Sentences

**We can throw in functions and equalities, too:**

- "Mike and Mary are the same age."
  - Are functional relations specified?
  - Are equalities specified?
  - **Answer:** `age(Mike) = age(Mary)`
- "There are exactly two shoes."
  - Are quantities specified?
  - Are equalities implied?
  - **Answer:** $\exists x\ \exists y\ shoe(x) \land shoe(y) \land \neg(x=y) \land \forall z\ (shoe(z) \Rightarrow (x=z) \lor (y=z))$

# Thinking in Logic Sentences

- We can use equality to define complex relations, such as *sibling(x, y)* in terms of simpler relations such as parent
  - "Siblings have the same parents."

$$
\begin{aligned}
\forall x\ \forall y\ \mathrm{sibling}(x,y) \Leftrightarrow \neg(x{=}y) \wedge \\
\exists m\ \exists f\ (\mathrm{parent}(m,x) \wedge \\
\mathrm{parent}(f,x) \wedge \\
\mathrm{parent}(m,y) \wedge \\
\mathrm{parent}(f,y))
\end{aligned}
$$

# Thinking in Logic Sentences

- Interesting words: **always, sometimes, never**
  - "Good people *always* have friends."

    $\forall x\ \mathtt{person}(x) \land \mathtt{good}(x) \Rightarrow \exists y(\mathtt{friend}(x,y))$
  - "Busy people *sometimes* have friends."

    $\exists x\ \mathtt{person}(x) \land \mathtt{busy}(x) \land \exists y(\mathtt{friend}(x,y))$
  - "Bad people *never* have friends."

    $\forall x\ \mathtt{person}(x) \land \mathtt{bad}(x) \Rightarrow \neg\exists y(\mathtt{friend}(x,y))$

# Thinking in Logic Sentences

- Interesting words: **always, sometimes, never**
  - "Good people *always* have friends."

    $\forall x\ \mathtt{person}(x) \wedge \mathtt{good}(x) \Rightarrow \exists y(\mathtt{friend}(x,y))$
  - "Busy people *sometimes* have friends."

    $\exists x\ \mathtt{person}(x) \wedge \mathtt{busy}(x) \wedge \exists y(\mathtt{friend}(x,y))$
  - "Bad people *never* have friends."

    $\forall x\ \mathtt{person}(x) \wedge \mathtt{bad}(x) \Rightarrow \neg\exists y(\mathtt{friend}(x,y))$

# Truth Tables

| P | Q | ~P | ~Q | P& Q | P V Q | P => Q |
|---|---|---|---|---|---|---|
| T | T | F | F | T | T | T |
| T | F | F | T | F | T | F |
| F | T | T | F | F | T | T |
| F | F | T | T | F | F | T |

# *Validity and Satisfiability*

- A sentence is valid if it is true in *all* models:

  $P_1 \lor \neg P_1$ $\qquad$ $P_1 \Rightarrow P_1$ (tautologies)

- A sentence is satisfiable if it is true in *some* models, or interpretations:

  $P_1 \lor P_2$

- A sentence is unsatisfiable if it is true in *no* models:

  $P_1 \land \neg P_1$ $\qquad$ (inconsistent/contradiction)

# *Entailment among sentences*

- If **S** is a set of sentences, and **p** is any sentence, we say that **p** is a logical consequence of **S**, or that **S** logically entails **p**, if and only if for every interpretation that satisfies **S** then the interpretation also satisfies **p**.

- In other words, **S** entails **p** iff every model of **S** satisfies **p**.
  - **S** |= **p**

- Note, that when **p** is valid, **S** can be empty.

# Proofs

- Truth tables allow us to assess entailment, but take an exponential amount of space and time.

- ***Proofs*** allow us to use a search procedure, such as DFS to derive a conclusion we are interested in.

- The states will be sets of propositions that are currently known to be true.

- The transitions for this search will add new propositions sanctioned by inference rules that have been shown to be valid (eg by a truth table). And we have lots of them....

# *Inference Rules (part 1)*

Modus Ponens $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$

And-Elimination (AE): $\dfrac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$

And-Introduction (AI): $\dfrac{\alpha_1,\ \alpha_2,\ \ldots\ ,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$

Or-Introduction (OI): $\dfrac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$

Double-Negation Elimination (DNE): $\dfrac{\neg\neg\alpha}{\alpha}$

Unit Resolution (UR): $\dfrac{\alpha \vee \beta,\ \neg\beta}{\alpha}$

Resolution (R): $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

deMorgan's Law (DML): $\dfrac{\neg(\neg\alpha \vee \beta)}{\alpha \wedge \neg\beta}$

Given the following knowledge base:

1. **P**
2. **P ⇒ R**
3. **R ⇒ ¬W**
4. **S ∨ R**
5. **(P ∧ R) ⇒ (S ∨ W)**

Prove **S** using natural deduction with these rules.

| | | |
|---|---|---|
| 6. | **R** | **(MP: 1,2)** |
| 7. | **¬W** | **(MP: 3,6)** |
| 8. | **P ∧ R** | **(AI: 1,6)** |
| 9. | **S ∨ W** | **(MP: 5,8)** |
| 10. | **S** | **(UR: 7,9)** |

# *More Inference Rules for FOL*

- **Universal Elimination, UE**

  variable substituted with ground term

  `∀x Eats(Jim, x)` infer `Eats(Jim, Cake)`

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

- **Existential Elimination, EE**

  variable substituted with *new* constant

  `∃x Eats(Jim, x)` infer `Eats(Jim, NewFood)`

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

- **Existential Introduction, EI**

  ground term substituted with variable

  `Eats(Jim, Cake)` infer `∃x Eats(x, Cake)`

$$\frac{\alpha}{\exists v\ \text{SUBST}(\{g/v\}, \alpha)}$$

# Unification

- Substitution $\theta$ is said to unify *p* and *q* if
  *SUBST($\theta$,p) = SUBST($\theta$,q)*

| *p* | *q* | $\theta$ |
|---|---|---|
| turtle(y) | turtle(Thom) | {y/Thom} |
| loves(Burr,x) | loves(Burr,Nat) | {x/Nat} |
| friends(Burr,x) | friends(y,Mark) | {y/Burr, x/Mark} |
| obeys(Ron,x) | obeys(z,mother(z)) | {z/Ron,x/mother(Ron)} |
| eats(y,y) | eats(z,Fish) | {y/z, z/Fish} |
| sees(JD,x,y) | sees(z,DJ,home(z)) | {z/JD,x/DJ,y/home(JD)} |
| sees(x,id(x), home(JD)) | sees(DJ,id(y),home(y)) | failure, assuming home(JD) ≠ home(DJ) |

# *Unification Algorithm*

- $\theta$ is a most general unifier (MGU)
  - Shortest length substitution list to make a match
  - In general, more than one MGU

- Our algorithm recursively explores the two expressions and simultaneously builds $\theta$

- We want to prevent replacing variables with terms that contains that variable (e.g. `{x/F(x)}`)
  - This slows down the algorithm

- Unification with this variable-substitution check has a time complexity of $O(n^2)$, where $n$ is the number of terms in the expressions

# *Generalized Modus Ponens (GMP)*

- Combines AI, UE, and MP into a single rule

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

**(where $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ for all $i$)**

- ***SUBST*** $(\theta, p)$ means apply substitutions in $\theta$ to sentence ***p***
- Substitution list □ = $\{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$ means
  - Replace all occurrences of variable $v_i$ with term $t_i$
  - Substitutions are made in left to right order

# Generalized Modus Ponens (GMP)

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

(where $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ for all $i$)

**Example:**

$p_1'$ = `taller(Larry,Curly)`

$p_2'$ = `taller(Curly,Moe)`

$p_1 \wedge p_2 \Rightarrow q$ = `taller(x,y)` $\wedge$ `taller(y,z)` $\Rightarrow$ `taller(x,z)`

$\theta$ = `{x/Larry, y/Curly, z/Moe}`

$SUBST(\theta, q)$ = `taller(Larry,Moe)`

# Generalized Modus Ponens (GMP)

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

# Generalized Modus Ponens (GMP)

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

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

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

$Owns(Nono, M_1)$ and $Missile(M_1)$

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

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

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 . . .

$$Enemy(Nono, America)$$

# *Forward Chaining with GMP*

**Step-1:**

In the first step we will start with the known facts and will choose the sentences which do not have implications. All these facts will be represented as below.

**American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)**

**Owns(*Nono*, M1) ... (2)**

**Missile(M1) ... (3)**

**Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)**

**Missile(x) ⇒ Weapons (x) ... (5)**

**Enemy(x, America) ⇒ Hostile(x) ... (6)**

**Enemy (*Nano*, America) ... (7)**

**American(West) ... (8)**

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

# *Forward Chaining with GMP*

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(*Nono*, M1) ... (2)

Missile(M1) ... (3)

Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)

Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

Enemy (*Nano*, America) ... (7)

American(West) ... (8)

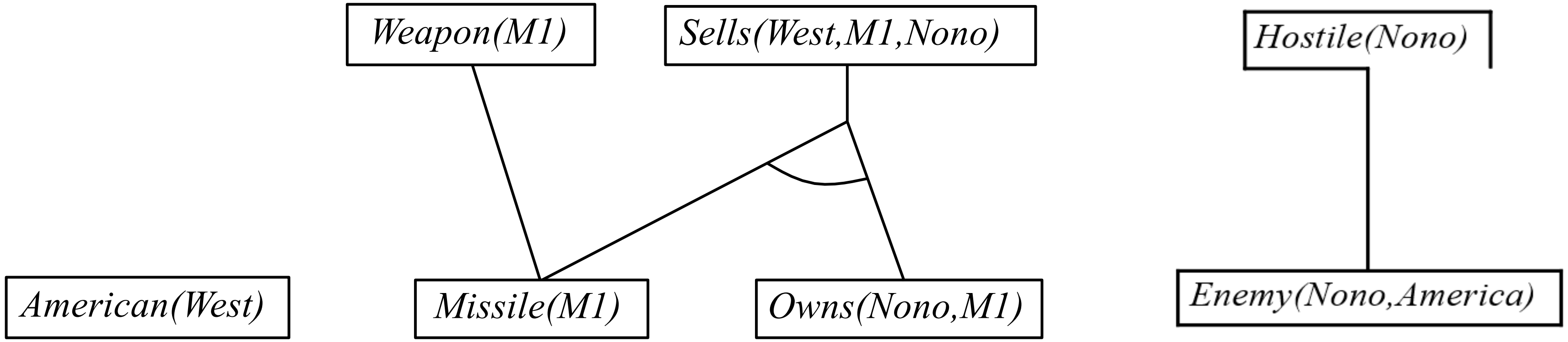
**Step-2:**

At the second step, we will see those facts which infer from available facts and with satisfied premises.

**Rule-(5)** satisfy with the **substitution {x/M1}, so Weapons(M11)** is added, and which infers from Rule(3).

**Rule-(4)** satisfy with the **substitution {x/M1}, so Sells (West, M1, Nono)** is added, which infers from the conjunction of Rule (2) and (3).

**Rule-(6)** is satisfied with the **substitution(x/Nano),** so Hostile(A) is added and which infers from Rule-(7).



# Forward Chaining with GMP

**Step-3:**

At step-3, as we can check Rule-(1) is satisfied with the substitution **{x/West, y/M1, z/Nano}, so we can add Criminal(West)** which infers all the available facts. And hence we reached our goal statement.

**Hence it is proved that West is Criminal using forward chaining approach.**

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(Nono, M1) ... (2)

Missile(M1) ... (3)

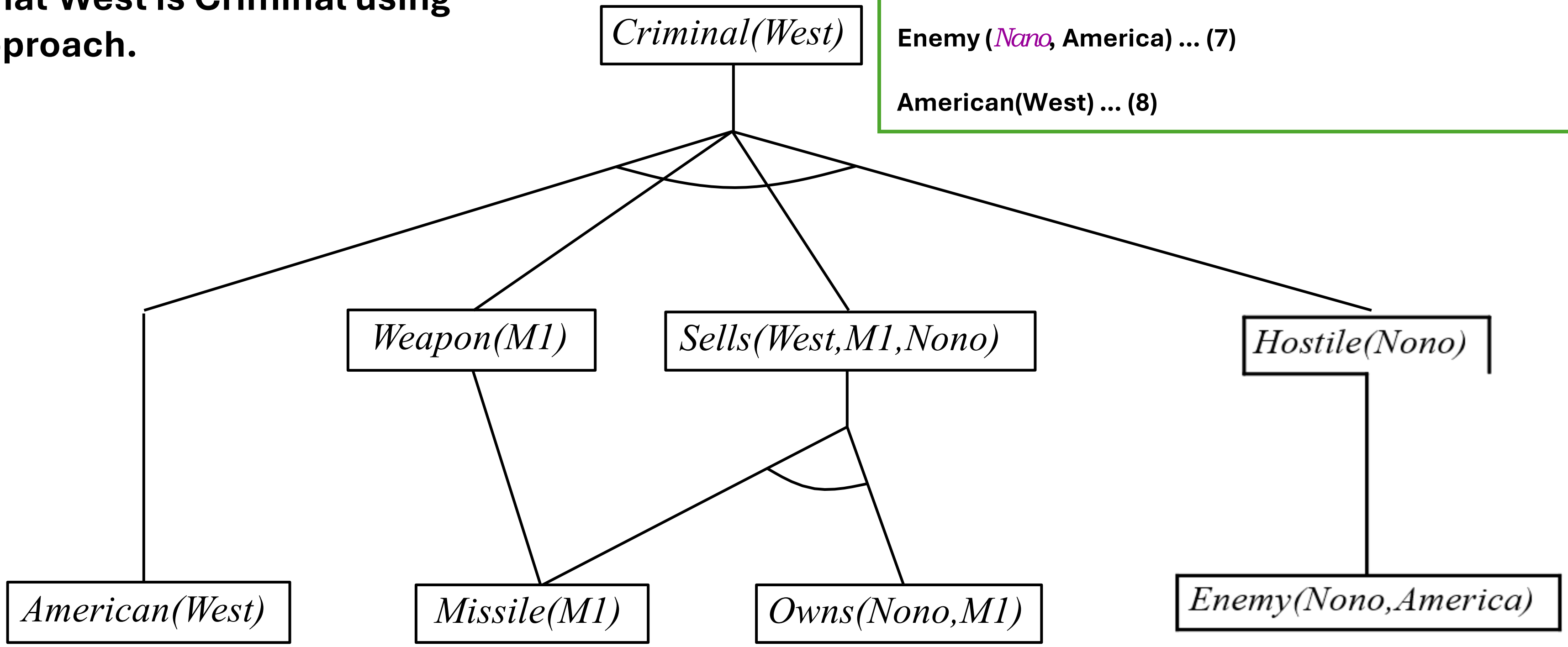
Missile(x) ∧ Owns(Nano, x) ⇒ Sells(West, x, Nano)... (4)

Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

Enemy (Nano, America) ... (7)

American(West) ... (8)



# Backward Chaining algorithm

- A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

- Is also known as a top-down approach, a backward deduction or backward reasoning method when using an inference engine.
  - It is also called a goal-driven approach, as a list of goals decides which rules are selected and used.

- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- The backward-chaining method mostly used a depth-first search strategy for proof.

# *Backward Chaining with GMP*

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(*Nono*, M1) ... (2)

Missile(M1) ... (3)

Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)

Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

Enemy (*Nano*, America) ... (7)

American(West) ... (8)

**Step-1:**

At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "**West is Criminal**," so following is the predicate of it.

$\boxed{Criminal(West)}$

# *Backward Chaining with GMP*

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(*Nono*, M1) ... (2)

Missile(M1) ... (3)

Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)

Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

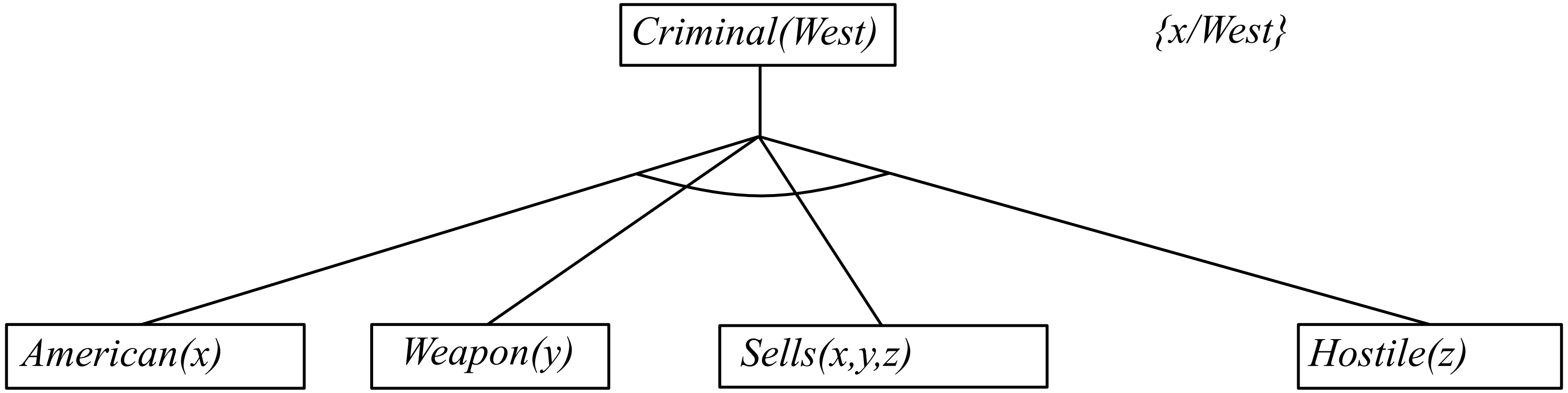
Enemy (*Nano*, America) ... (7)

American(West) ... (8)

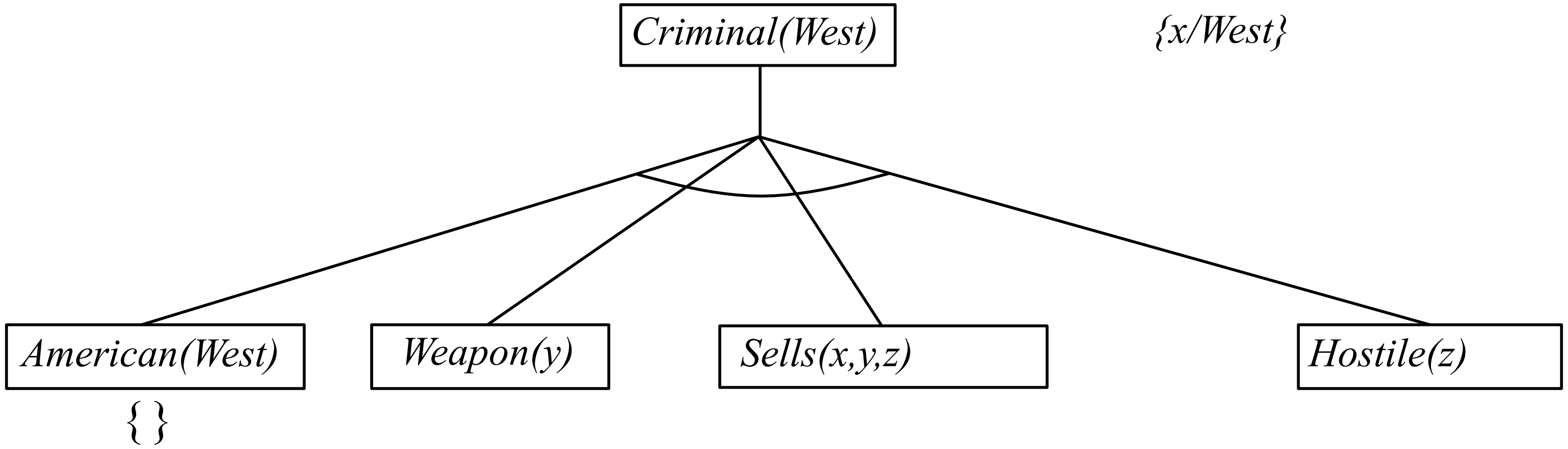
**Step-2:**

At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate **Criminal (West)** is present with substitution **{West/x}.** So we will add all the conjunctive facts below the first level and will replace x with **West**.

**Here we can see American (West) is a fact, so it is proved here**



# Backward Chaining with GMP



# Backward Chaining with GMP

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(*Nono*, M1) ... (2)

Missile(M1) ... (3)

Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)

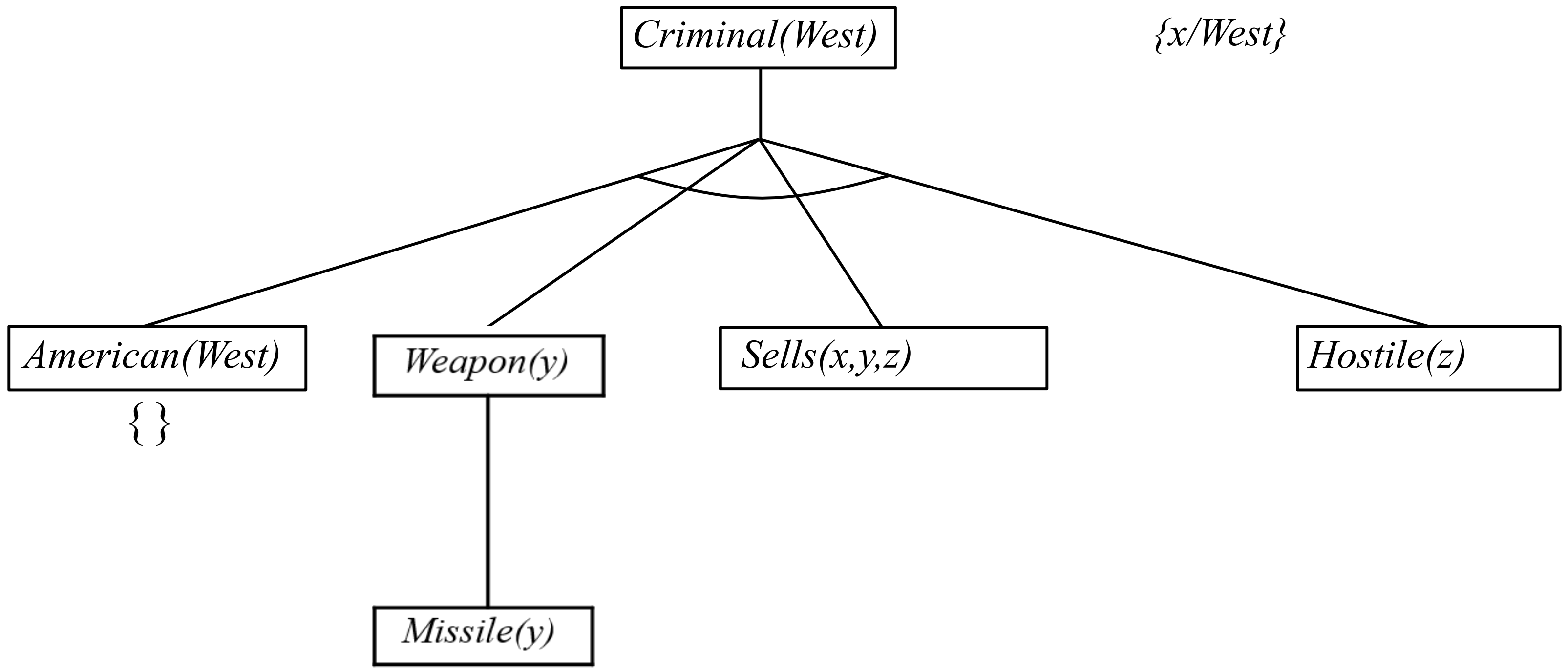
Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

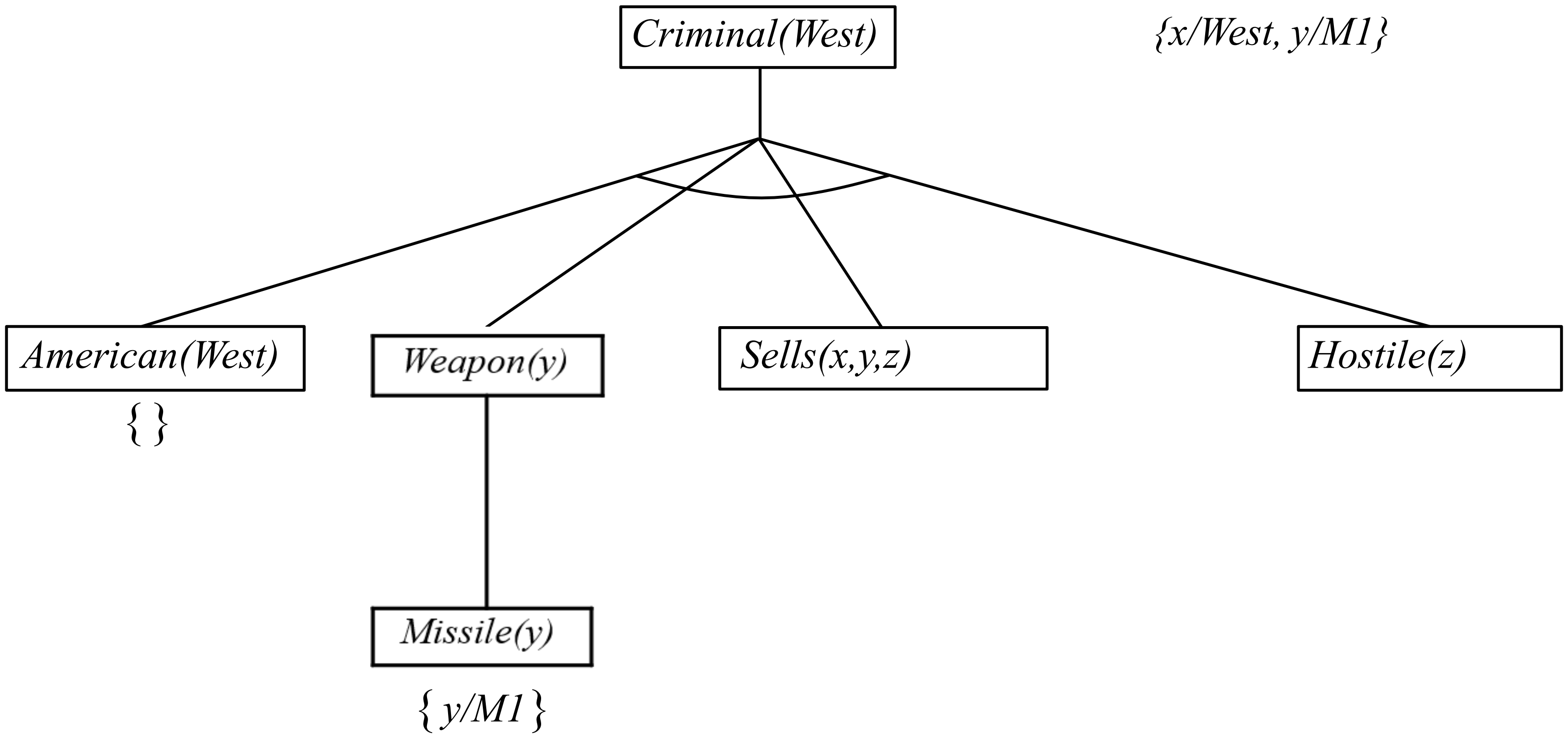
Enemy (*Nano*, America) ... (7)

American(West) ... (8)

**Step-3:**t At step-3, we will extract further fact **Missile(y)** which infer from **Weapon(y)**, as it satisfies Rule-(5). Weapon (y) is also true with the substitution of a constant **M1** at **y**.



# Backward Chaining with GMP



# Backward Chaining with GMP

**Step-4:**t At step-4, we can infer facts Missile(T1) and Owns(A, T1) form **Sells(West, M1, z)** which satisfies the **Rule- 4**, with the substitution of **Nano** in place of **z**. So these two statements are proved here.

**American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)**

**Owns(*Nono*, M1) ... (2)**

**Missile(M1) ... (3)**

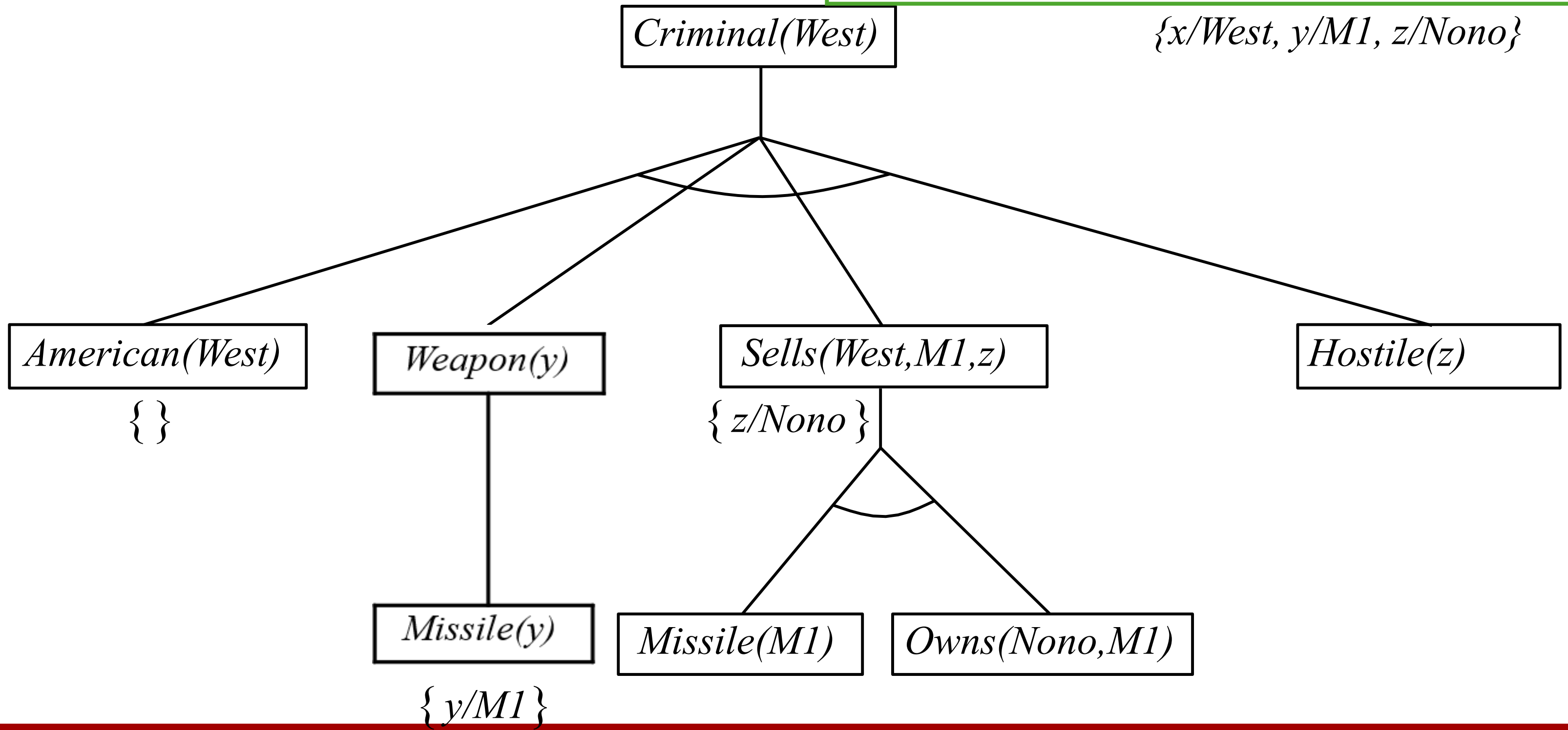
**Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)**

**Missile(x) ⇒ Weapons (x) ... (5)**

**Enemy(x, America) ⇒ Hostile(x) ... (6)**

**Enemy (*Nano*, America) ... (7)**

**American(West) ... (8)**



# Backward Chaining with GMP

**Step-5:**t At step-5, we can infer the fact Enemy(Nano, America) from Hostile(Nano) which satisfies Rule-6. And hence all the statements are proved true using backward chaining.

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) ... (1)

Owns(*Nono*, M1) ... (2)

Missile(M1) ... (3)

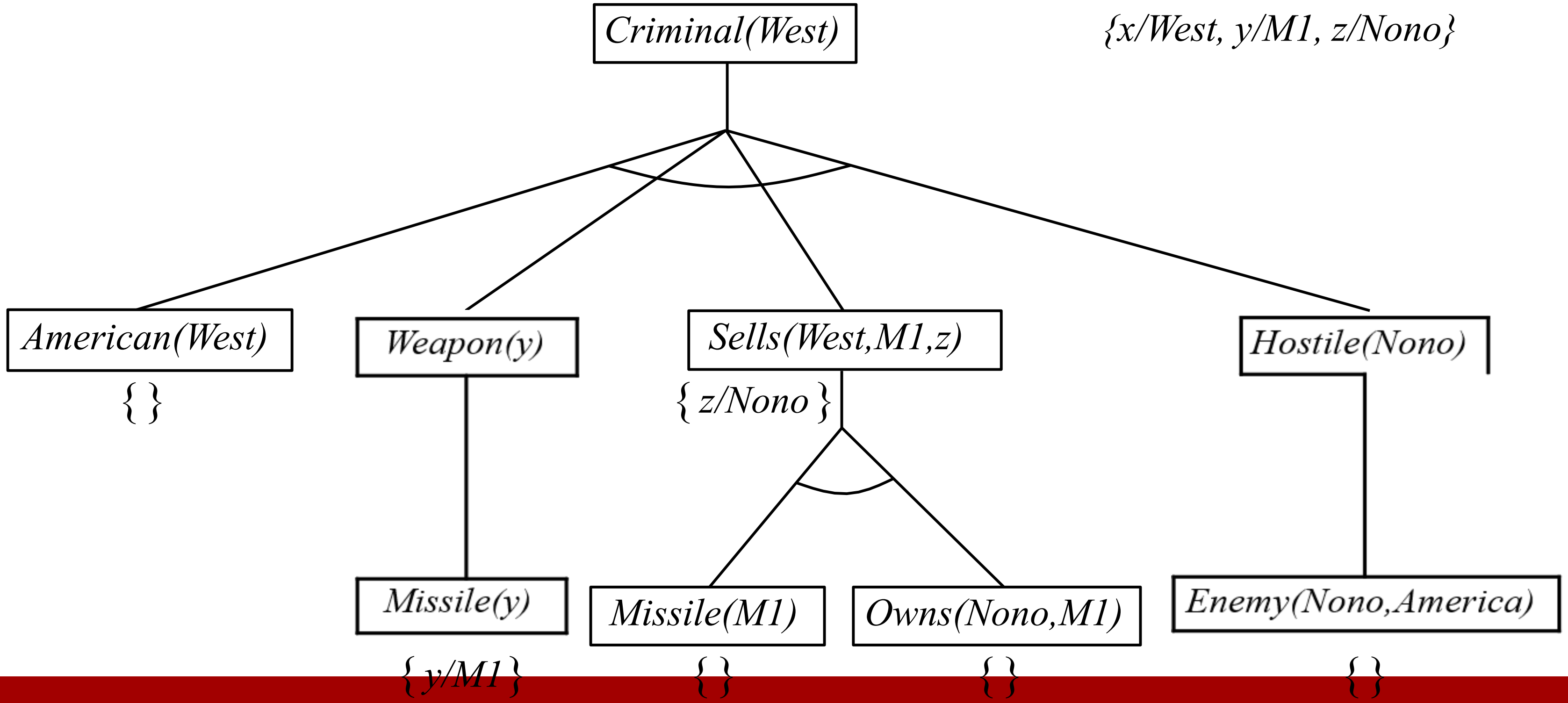
Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)... (4)

Missile(x) ⇒ Weapons (x) ... (5)

Enemy(x, America) ⇒ Hostile(x) ... (6)

Enemy (*Nano*, America) ... (7)

American(West) ... (8)



# *Backward Chaining with GMP*

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal (x) … (1)

Owns(*Nano*, M1) … (2)

Missile(M1) … (3)

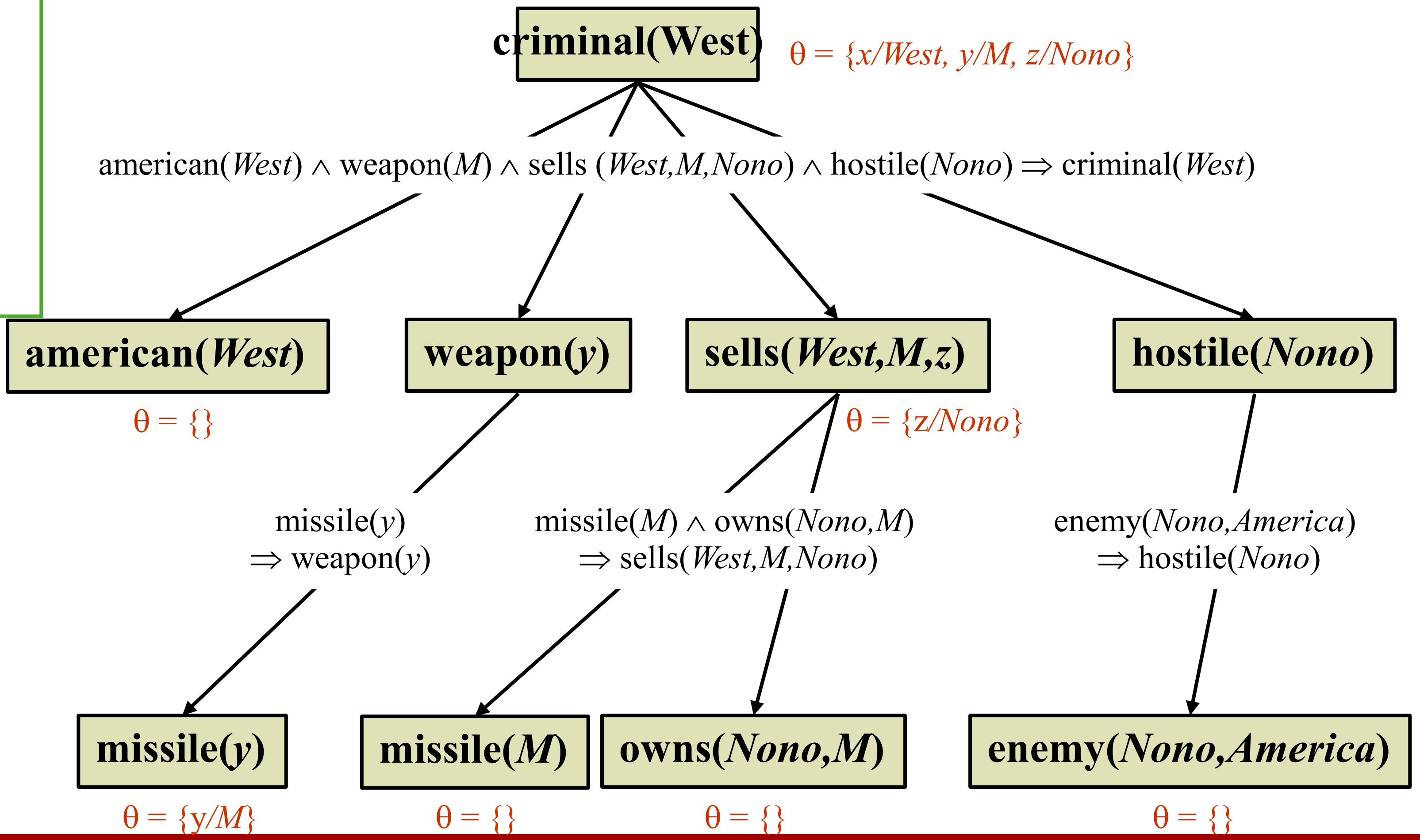
Missile(x) ∧ Owns(*Nano*, x) ⇒ Sells(West, x, *Nano*)… (4)

Missile(x) ⇒ Weapons (x) … (5)

Enemy(x, America) ⇒ Hostile(x) … (6)

Enemy (*Nano*, America) … (7)

American(West) … (8)



# *Forward vs Backward Chaining*

**Forward:**

- Sound and complete for first-order definite clauses (ie horn normal form)
- Adds all sentences that can be inferred (
  - Matching premises against known facts in NP-Hard
- Often used  in deductive databases

**Backward:**

- DFS so space is linear in the size of the proof
- Incomplete if infinite loops
  - Fix by checking current goal against every goal on the stack
- Inefficient due to repeated subgoals
  - Fix by caching previous results (but this uses up space!)
- Used for logic programming systems (e.g. Prolog)

# *Resolution*

- Resolution is a refutation technique:
  - To prove $\mathbf{KB} \models \boldsymbol{\alpha}$ show that $\mathbf{KB} \wedge \neg \boldsymbol{\alpha}$ is unsatisfiable
- Resolution uses $\mathbf{KB}$ and $\neg \boldsymbol{\alpha}$ in CNF:
  - Conjunction of clauses that are disjunction of literals
- Entailment in general FOL is only semi-decidable:
  - Can prove $\boldsymbol{\alpha}$ if $\mathbf{KB} \models \boldsymbol{\alpha}$
  - Cannot always prove that $\mathbf{KB}$ doesn't $\models \boldsymbol{\alpha}$

✱ *Resolution repeatedly combines two clauses to make a new one until an empty clause is derived (a contradiction)*

# Resolution Refutation

$$\frac{\texttt{well-fed(Me)},\ \neg\texttt{well-fed(x)} \lor \texttt{happy(x)}}{SUBST(\theta, \texttt{happy(x)})}$$

- $p_j$ is `well-fed(Me)`
  $q_k$ is $\neg$`well-fed(x)`

- ***UNIFY***$(p_j, q_k)$ result in $\theta$ = `{x/Me}`
  ***SUBST***(x/Me, **happy(x)**) result in `happy(Me)`

  **Inferred sentence:** `happy(Me)` ⇐**RESOLVENT**

∗ ***GMP is a special case of generalized resolution (for KBs in HNF)***

# Resolution Refutation Example

Recycling the "West is a criminal" example, let's begin by making sure that all the facts and rules in our KB are in CNF. The following are already in CNF:

$\mathbf{enemy}(Nono, America)$ $\quad$ $\mathbf{owns}(Nono, M)$

$\mathbf{missile}(M)$ $\quad$ $\mathbf{american}(West)$

The remaining four need to be converted to CNF:

$\mathbf{american}(x) \wedge \mathbf{weapon}(y) \wedge \mathbf{sells}\ (x,y,z) \wedge \mathbf{hostile}(z) \Rightarrow \mathbf{criminal}(x)$

$\neg\mathbf{american}(x) \vee \neg\mathbf{weapon}(y) \vee \neg\mathbf{sells}\ (x,y,z) \vee \neg\mathbf{hostile}(z) \vee \mathbf{criminal}(x)$

$\mathbf{missile}(x) \wedge \mathbf{owns}(Nono,x) \Rightarrow \mathbf{sells}(West,x,Nono)$

$\neg\mathbf{missile}(x) \vee \neg\mathbf{owns}(Nono,x) \vee \mathbf{sells}(West,x,Nono)$

$\mathbf{enemy}(x,America) \Rightarrow \mathbf{hostile}(x)$

$\neg\mathbf{enemy}(x,America) \vee \mathbf{hostile}(x)$

$\mathbf{missile}(x) \Rightarrow \mathbf{weapon}(x)$

$\neg\mathbf{missile}(x) \vee \mathbf{weapon}(x)$

And we also *first* need to negate our query: $\neg$ **criminal(*West*)**

# *Completed Refutation Example*

$\text{enemy}(Nono, America)$

$\text{owns}(Nono, M)$

$\text{missile}(M)$

$\text{american}(West)$

$\neg\text{american}(x) \lor \neg\text{weapon}(y) \lor \neg\text{sells}(x,y,z) \lor \neg\text{hostile}(z) \lor \text{criminal}(x)$

$\neg\text{missile}(x) \lor \neg\text{owns}(Nono,x) \lor \text{sells}(West,x,Nono)$

$\neg\text{enemy}(x,America) \lor \text{hostile}(x)$

$\neg\text{missile}(x) \lor \text{weapon}(x)$

And we also *first* need to negate our query: $\neg$ **criminal(*West*)**

$\neg\text{american}(x) \lor \neg\text{weapon}(y) \lor$
$\neg\text{sells}(x,y,z) \lor \neg\text{hostile}(z) \lor \text{criminal}(x)$

$\neg\text{criminal}(West)$

$\neg\text{american}(West) \lor \neg\text{weapon}(y) \lor$
$\neg\text{sells}(West,y,z) \lor \neg\text{hostile}(z)$

$\text{american}(West)$

$\neg\text{weapon}(y) \lor \neg\text{sells}(West,y,z) \lor \neg\text{hostile}(z)$

$\neg\text{missile}(x) \lor \text{weapon}(x)$

$\neg\text{missile}(y) \lor \neg\text{sells}(West,y,z) \lor \neg\text{hostile}(z)$

$\text{missile}(M)$

$\neg\text{sells}(West,M,z) \lor \neg\text{hostile}(z)$

$\neg\text{missile}(x) \lor \neg\text{owns}(Nono,x) \lor$
$\text{sells}(West,x,Nono)$

$\neg\text{missile}(M) \lor \neg\text{owns}(Nono,M) \lor \neg\text{hostile}(Nono)$

$\text{missile}(M)$

$\neg\text{owns}(Nono,M) \lor \neg\text{hostile}(Nono)$

$\text{owns}(Nono,M)$

$\neg\text{hostile}(Nono)$

$\neg\text{enemy}(x,America) \lor \text{hostile}(x)$

$\neg\text{enemy}(Nono,America)$

$\text{enemy}(Nono,America)$

$\square$

# Answer extraction

- Although resolution is good at answering yes-no queries, it does not tell us what entity makes the query true.

- One solution, called answer extraction, involves replacing a query $\exists y\, P(y)$ with by $\exists y\, P(y)\ \&\ \neg A(x)$ where A is a new predicate symbol occurring nowhere else that we call the **answer predicate**.

- Since A occurs nowhere else, it will not be possible to derive the empty clause and we can terminate the derivation as soon as the resolvent contains only the answer predicate.

- The binding for x in A(x) will be the answer we want.

# Completed Refutation Example with answer extraction

¬criminal(*A*) V Answer (A)

¬american(*x*) ∨ ¬weapon(*y*) ∨ ¬sells (*x*,*y*,*z*) ∨ ¬hostile(*z*) ∨ criminal(*x*)

¬american(A) ∨ Answer(A) ∨ ¬weapon(*y*) ∨ ¬sells(*A*,*y*,*z*) ∨ ¬hostile(*z*)

american(West)

¬weapon(*y*) ∨ Answer(West) ∨ ¬sells(*West*,*y*,*z*) ∨ ¬hostile(*z*)

¬missile(*x*) ∨ weapon(*x*)

¬missile(*y*) ∨ Answer(West) ∨ ¬sells(*West*,*y*,*z*) ∨ ¬hostile(*z*)

missile(*M*)

¬sells(*West*,*M*,*z*) ∨ Answer(West) ∨ ¬hostile(*z*)

¬missile(*x*) ∨ ¬owns(*Nono*,*x*) ∨ sells(*West*,*x*,*Nono*)

¬missile(*M*) ∨ Answer(West) ∨ ¬owns(*Nono*,*M*) ∨ ¬hostile(*Nono*)

missile(*M*)

¬owns(*Nono*,*M*) ∨ Answer(West) ∨ ¬hostile(*Nono*)

owns(*Nono*,*M*)

¬hostile(*Nono*) ∨ Answer(West)

¬enemy(*x*,*America*) ∨ hostile(*x*)

¬enemy(*Nono*,*America*) ∨ Answer(West)

enemy(*Nono*,*America*)

□ Answer(West)

# *Converting FOL to CNF to allow Resolution*

1. Replace $\Leftrightarrow$ with equivalent: $\mathbf{P} \Leftrightarrow \mathbf{Q}$ **becomes** $\mathbf{P} \Rightarrow \mathbf{Q} \wedge \mathbf{Q} \Rightarrow \mathbf{P}$
2. Replace $\Rightarrow$ with equivalent: $\mathbf{P} \Rightarrow \mathbf{Q}$ **becomes** $\neg\mathbf{P} \vee \mathbf{Q}$
3. Reduce scope of $\neg$ to single literals:

   $\neg\neg\mathbf{P}$ **becomes** $\mathbf{P}$ (DNE)

   $\neg(\mathbf{P} \vee \mathbf{Q})$ **becomes** $\neg\mathbf{P} \wedge \neg\mathbf{Q}$ (de Morgan's)

   $\neg(\mathbf{P} \wedge \mathbf{Q})$ **becomes** $\neg\mathbf{P} \vee \neg\mathbf{Q}$ (de Morgan's)

   $\neg\forall\mathbf{x}\ \mathbf{P}$ **becomes** $\exists\mathbf{x}\ \neg\mathbf{P}$

   $\neg\exists\mathbf{x}\ \mathbf{P}$ **becomes** $\forall\mathbf{x}\ \neg\mathbf{P}$

4. Standardize variables apart:
   - Each quantifier must have a unique variable name
   - Avoids confusion in steps 5 and 6
   - *e.g.* $[\forall\mathbf{x}\ \mathbf{P}] \vee [\exists\mathbf{x}\ \mathbf{Q}]$ **becomes** $\forall\mathbf{x}\ \mathbf{P} \vee \exists\mathbf{y}\ \mathbf{Q}$

5. Eliminate existential quantifiers (Skolemize):

$\exists\ x\ P(x)$ **becomes** $P(K)$ (EE)

**K** is some new constant (Skolem constant)

- *e.g.* $\forall x \exists y\ P(x,y)$ **becomes** $\forall x\ P(x,F(x))$

  **F()** must be a new function (Skolem function) with arguments that are all enclosing universally quantified variables

- Everyone has a name.

  $\forall x\ person(x) \Rightarrow \exists y\ name(y) \land has(x,y)$

  **wrong:** $\forall x\ person(x) \Rightarrow name(K) \land has(x,K)$

  Everyone has the same name K!!

  We want everyone to have a name based on who they are

  **right:** $\forall x\ person(x) \Rightarrow name(F(x)) \land has(x,F(x))$

# Converting FOL to CNF

6. Drop universal quantifiers:

All variables are only universally quantified after step 5

*e.g.* $\forall x\, P(x) \vee \forall y\, Q(y)$ **becomes** $P(x) \vee Q(y)$

All variables in KB will be assumed to be universally quantified

7. Distribute $\vee$ over $\wedge$ :

$(P \wedge Q) \vee R$ **becomes** $(P \vee R) \wedge (Q \vee R)$

8. Group conjunctions/disjunctions together:

$(P \wedge Q) \wedge R$ **becomes** $(P \wedge Q \wedge R)$

$(P \vee Q) \vee R$ **becomes** $(P \vee Q \vee R)$

# *Example of PL: resolution*

**Let the KB be:**

**R1:** bird

**R2:** bird -> flies

**R3:** flies -> hasWings

α: **hasWings**

**To prove $KB \models \alpha$ show that $KB \wedge \neg \alpha$ is unsatisfiable**

- Transformation of **KB** and **¬α** into CNF:

**R1:** bird

**R2:** $\neg$ bird $\vee$ flies

**R3:** $\neg$ flies $\vee$ hasWings

**¬α:** $\neg$ hasWings

# Example of PL: resolution

**Resolution tree:**

$$
\begin{tikzcd}
\neg hasWings \arrow[d] & \neg flies \vee hasWings \arrow[dl] \\
\neg flies \arrow[d] & \neg bird \vee flies \arrow[dl] \\
\neg bird \arrow[d] & bird \arrow[dl] \\
\square &
\end{tikzcd}
$$

# *Example of FOL or FOPC: resolution*

Consider the following sentences:

- *Tweety is a bird.*
- *All birds can fly.*
- *Everyone who can fly has wings.*

*Give the representation of these statements in FOL (FOPC). Using the resolution principle shows that these sentences imply*

- ***" Tweety has wings."***

# *Example of FOL or FOPC: resolution*

## Representation in FOL:

- *Tweety is a bird.*
  - R1: Bird(Tweety)
- *All birds can fly.*
  - R2: $\forall x.\ Bird(x) \rightarrow Flies(x)$
- *Everyone who can fly has wings.*
  - R3: $\forall x.\ Flies(x) \rightarrow HasWings(x)$
- ***" Tweety has wings."***
  - α: HasWings(Tweety).

# *Example of FOL or FOPC: resolution*

To prove **KB** $\models$ **α** show that $\mathbf{KB} \land \neg \boldsymbol{\alpha}$ is unsatisfiable :

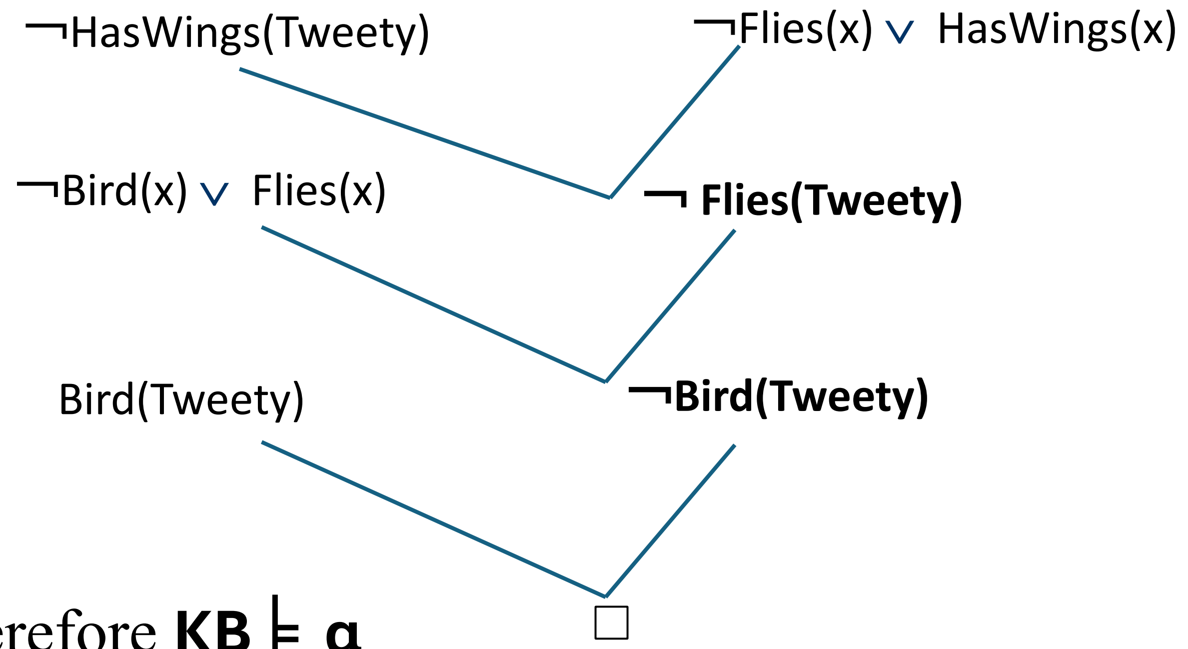
1. Transformation of **KB** and **¬α** into CNF:
   - Bird(Tweety)
   - ¬Bird(x) ∨ Flies(x)
   - ¬Flies(x) ∨ HasWings(x)
   - ¬α :¬HasWings(Tweety).

# *Example of FOL or FOPC: resolution*

To prove **KB** ⊨ **α** show that **KB** ∧ ¬ ***α*** is unsatisfiable :

1. Transformation of **KB** and **¬α** into CNF:
   - R1: Bird(Tweety)
   - R2: ¬Bird(x) ∨ Flies(x)
   - R3: ¬Flies(x) ∨ HasWings(x)
   - ¬α : ¬HasWings(Tweety).

**Derivation tree**




Therefore **KB** ⊨ **α**