UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Artificial Intelligence

First-Order Logic: Inference

LESSON 15

prof. Antonino Staiano

M.Sc. In "Machine Learning e Big Data" - University Parthenope of Naples

Inference in First-Order Logic

- Inference algorithms are more complex than in propositional logic, due to quantifiers and functions
- Basic tools: two inference rules for sentences with quantifiers (Universal and Existential Instantiation), that derive sentences without quantifiers
- This reduces first-order inference to propositional inference, with complete but semi-decidable inference procedures:
  - algorithms exist that find a proof KB $\vdash \alpha$ in a finite number of steps for every entailed sentence KB $\models \alpha$
  - no algorithm is capable to find the proof KB $\nvdash \alpha$ in a finite number of steps for every non-entailed sentence KB $\not\models \alpha$
- Therefore, since one does not know that a sentence is entailed until the proof is done, when a proof procedure is running one does not know whether it is about to find proof, or whether it will never find one

Inference in FOL

- Modus Ponens can be generalized to FOL, leading to the first-order versions of the Forward Chaining and Backward Chaining algorithms, which are complete and semi-decidable and limited to Horn clauses
- The Resolution rule can also be generalized to predicate logic, leading to the first-order version of the complete but semi-decidable resolution algorithm

Inference: Propositional vs First-Order

- A first approach for inference in FOL is to convert the FOL KB to PL
  - Apply propositional inference
- As an example, let's suppose that the KB contains the axiom that all greedy kings are evil
  - $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - we can infer
    - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
    - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
    - $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$
- The previous example, suggests that we can apply a general rule for Universal instantiation (UI)
  - *Any sentence we get by substituting a ground term, i.e., a term with no variables, for a universally quantified variable can be* ***inferred***

Eliminating Universal Quantifiers

- Let θ denote a substitution list $\{v_1/t_1, \ldots, v_n/t_n\}$, where:
  - $v_1,\ldots,v_n$ are variable names
  - $t_1, \ldots, t_n$ are terms (either constant symbols, variables, or functions recursively applied to terms)
- Let SUBST(θ, α) denote the sentence obtained by applying the substitution θ to the sentence α
  - Example:
    - SUBST({y/One}, ∀x, y Eq(S(x), S(y)) ⇒ Eq(x, y))

    produces

    ∀x Eq(S(x), S(One)) ⇒ Eq(x, One)

Universal Instantiation

- The rule is

$$\frac{\forall \mathrm{v}\ \alpha}{\mathrm{SUBST}(\{\mathrm{v/t}\ \},\alpha)}$$

  where t can be any ground term, and v any variable

- Since a sentence ∀x α[x] states that α is true for every domain element in place of x, then one can derive that α is true for any given element t
- Example: from ∀x N(x) ⇒ N(S(x)) one can derive
  - N(Z) ⇒ N(S(Z)), for θ = {x/Z}
  - N(S(S(Z))) ⇒ N(S(S(S(Z)))), for θ = {x/S(S(Z))}
  - …

Existential Instantiation

- The same approach can be derived for eliminating the existential quantifier
- It is called Existential Instantiation (EI)

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

where t must be a new constant symbol, called Skolem constant, that does not appear elsewhere in the KB

- A sentence $\exists v\ \alpha[v]$ states that there is some domain element satisfying a condition
  - The above rule just gives a name to one such an element, but that name must not belong to another element
- Example
  - $\exists x\ Crown(x) \wedge OnHead(x,John)$
    - $Crown(C1) \wedge OnHead(C1, John)$, provided that *C1* is not in the KB

Existential Instantiation

- UI can be applied several times to add new sentences
  - The new KB is logically equivalent to the old one
- EI can be applied once to replace the existential sentence
  - The new KB is not logically equivalent to the old one, but it is inferentially equivalent
    - it is satisfiable iff the old KB was satisfiable

Exercise: Existential Instantiation

- Suppose a knowledge base contains just one sentence
  - ∃x *AsHighAs*(x,Everest)
- Which of the following are legitimate results of applying Existential Instantiation?

  a. *AsHighAs*(Everest, Everest)
  b. *AsHighAs*(Kilimanjaro, Everest)
  c. *AsHighAs*(Kilimanjaro, Everest) ∧ AsHighAs(BenNevis, Everest) (after two applications)

Inference Algorithms and Quantifiers

- First-order inference algorithms usually apply Existential Instantiation as a pre-processing step
  - every existentially quantified sentence is replaced by a single sentence
  - Accordingly, the resulting KB contains only sentences without variables and sentences where all the variables are universally quantified
- Another useful pre-processing step is renaming all the variables in the KB to avoid name clashes between variables used in different sentences
  - For instance, the variables in $\forall x\ P(x)$ and $\forall x\ Q(x)$ are not related to each other, and renaming any of them (say, $\forall y\ Q(y)$) produces an equivalent sentence

Reduction to Propositional Inference

- Propositionalization
  - Suppose the KB contains just the following sentences and facts
    - $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$Evil(x)
    - King(John)
    - Greedy(John)
    - Brother(Richard, John)
  - Instantiating the universal sentence in all possible substitutions {x/John}, {x/Richard} (pretending that John and Richard are the only objects), we have
    - King(John) $\wedge$ Greedy(John) $\Rightarrow$Evil(John)
    - King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$Evil(Richard)
    - King(John)
    - Greedy(John)
    - Brother(Richard, John)
  - Next, replace atomic sentences with proposition symbols
    - That is, King(John) with KingJohn
    - And apply any of the propositional inference algorithm

Reduction to Propositional Inference

- When the KB includes function symbols, a problem emerges
  - there are possibly infinitely ground-term substitutions
  - E.g., Father(Father(Father(John)))
- However, a theorem due to Herbrand (1930) states
  - *If a sentence α is entailed by a FOL KB, it is entailed by a finite subset of the propositional KB*
- … and suggests

```
for n in range(∞)
    Create a propositional KB by instantiating with depth-n terms
    See if this KB ⊨ α
```

  - It works if KB ⊨ α, loops otherwise
    - That is, entailment in FOL is semidecidable
      - No algorithm exists that says no to every non-entailed sentence

Problems with Propositionalization

- Propositionalization could generate lots of irrelevant sentences
- For instance, from
  - $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$Evil(x)
  - King(John)
  - $\forall$ y Greedy(y)
  - Brother(Richard, John)
  - it seems obvious that Evil (John), but propositionalization produces lots of irrelevant facts such as Greedy (Richard)
- With *pk-ary* predicates and n constants, there are pn^k instantiations
  - With function symbols, it gets much worse!

Generalized Modus Ponens

- The aim is to use a single rule to reason that {x/John} solves the query Evil(x)
- As an example, given that $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$Evil(x)
  - find some x such that x is a king and x is greedy, then this x is evil
    - i.e., if there is some substitution θ that makes the conjuncts of the premise identical to sentences in the KB, then we can assert the conclusion after applying θ
  - {x/John} does the job
- If the KB contains $\forall$ y Greedy(y) instead of Greedy(John) then we want still be able to conclude Evil(John), since King(John) and John is greedy (since everyone is greedy)
  - we need to find a substitution for both variables x and y that makes the premises identical to sentences in the KB
    - {x/John, y/John} does the job

Generalized Modus Ponens

- Generalized Modus Ponens (GMP):
  - given atomic sentences (non-negated predicates) p_i , p'_i , i = 1,...,n, and q, and a substitution θ such that SUBST(θ,p_i) = SUBST(θ, p'_i) for all i:

$$\frac{(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q), \quad p'_1, p'_2 \ldots, p'_n}{\text{SUBST}(\theta, q)}$$

- GMP is sound
  - It must be shown that $p'_1\, p'_2 \ldots p'_n$, $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models$ SUBST(θ,q),
    - provided that SUBST(θ,p_i) = SUBST(θ, p'_i) for all i
  - Lemma: For any definite clause p, we have p$\models$ SUBST(θ,p) by UI

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models$ SUBST(θ, $p_1 \wedge \ldots \wedge p_n \Rightarrow q$) = SUBST($\theta$, p_1) $\wedge \ldots \wedge$ SUBST(θ, p_n) $\Rightarrow$ SUBST(θ,q)
2. $p'_1\, p'_2 \ldots p'_n \models p'_1 \wedge p'_2 \wedge \ldots \wedge p'_n \models$ SUBST(θ, p'_1) $\wedge \ldots \wedge$ SUBST(θ, p'_n)
3. From 1 and 2, SUBST(θ,q) follows by ordinary Modus Ponens

Unification

- A widely used tool in first-order inference algorithms is unification, the process of finding a substitution (if any) that makes two sentences (where at least one contains variables) identical
- For instance, ∀x,y *Knows*(x,y) and ∀z *Knows*(*John*,z) can be unified by different substitutions
  - Assuming that Bill is one of the constant symbols, two possible unifiers are:
    - {x/John,y/Bill,z/Bill}
    - {x/John,y/z}
- Among all possible unifiers, the one of interest for first-order inference algorithms is the most general unifier, i.e., the one that places the fewest restrictions on the values of the variables
  - The only constraint is that every occurrence of a given variable can be replaced by the same term
- In the above example, the most general unifier is {x/John,y/z}, as it does not restrict the value of y and z

Unification Example

- Consider the sentence $\forall x$ *Knows(John,x)* (*John knows everyone*). Assume that the KB also contains the following sentences (note that different variables names are used in different sentences):
  1. *Knows(John,Jane)*
  2. $\forall y$ *Knows(y,Bill)*
  3. $\forall z$ *Knows(z,Mother(z))*
  4. *Knows(Elizabeth,Bill)*
- The most general unifier with *Knows(John,x)* is:
  1. *{x/Jane}*
  2. *{y/John,x/Bill}*
  3. *{z/John,x/Mother(John)}*
  4. no unifier exists, as the constant symbols *John* and *Elizabeth* in the first argument are different

Exercise: Unification

- For each pair of atomic sentences, give the most general unifier if it exists:
  - *P*(A, B, B), *P*(x, y, z)
    {x/A, y/B, z/B} (or some permutation of this)
  - *Q*(y, G(A,B)),*Q*(G(x,x), y)
    No unifier (x cannot bind to both A and B)
  - *Older*(Father(y), y),*Older*(Father(x), John)
    {y/John,x/John}
  - *Knows*(Father(y), y),*Knows*(x, x)
    No unifier (because the occurs-check prevents unification of y with Father(y))

Propositionalization vs GMP

- Consider again the domain made up of two individuals denoted with the constant symbols *John* and *Richard*, and the following KB:
  1) $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  2) $\forall y\ Greedy(y)$
  3) $King\ (John)$
  4) $Brother\ (Richard\ ,\ John)$
- Intuitively, this entails *Evil(John)*, i.e., $KB \models Evil(John)$
- The corresponding inference $KB \vdash Evil(John)$ can be obtained by using the inference rules after propositionalization, as shown in the following

Propositionalization vs GMP

- Applying Universal Instantiation to (1) produces:
  5) *King(John) ∧ Greedy(John) ⇒ Evil(John), with {x/John}*
  6) *King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard), with {x/Richard}*
- Applying Universal Instantiation to (2, i.e., *∀y Greedy(y)*) produces:
  7) *Greedy(John), with {y/John}*
  8) *Greedy(Richard), with {y/Richard}*
- Applying And Introduction to (3) and (7) produces:
  9) *King(John)∧Greedy(John)*
- Applying Modus Ponens to (5) and (9) produces:
  10) *Evil(John)*

Propositionalization vs GMP

- All but the last inference steps in the above example can be seen as pre-processing steps whose aim is to "prepare" the application of Modus Ponens
- Moreover, some of these steps (Universal Instantiation using the symbol Richard) are clearly useless to derive the consequent of implication (1)
  - i.e., Evil(John)
- In the previous example, GMP allows *Evil(John)* to be derived in a single step
  - and avoids unnecessary applications of inference rules like Universal Instantiation to sentences (1) and (2) with *{x/Richard}* or *{y/Richard}*
- GMP can be applied to sentences (1), (2), and (3), with θ = *{x/John, y/John}*
  1) $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  2) $\forall y\ Greedy(y)$
  3) $King\ (John)$
  - this immediately derives *Evil(John)*
- The key advantage of GMP inference rule over propositionalization is that it makes only those substitutions that are required to allow inferences to proceed

Horn Clauses in FOL

- GMP allows the Forward Chaining (FC) and Backward Chaining (BC) inference algorithms to first-order logic
  - This in turn requires the concept of Horn clause
- A Horn clause in FOL is an implication $\alpha \Rightarrow \beta$ in which:
  - α is a conjunction of non-negated predicates
  - β is a single non-negated predicate
  - all variables (if any) are universally quantified, and the quantifier appears at the beginning of the sentence
- Example: $\forall x\ (P(x) \wedge Q(x)) \Rightarrow R(x)$
- Also, single predicates (possibly negated) are Horn clauses:
  - $P(t_1,\ldots,t_n)$ equivalent to $(\text{True} \Rightarrow P(t_1,\ldots,t_n))$
  - $\neg P(t_1, \ldots, t_n)$ equivalent to $(P(t_1, \ldots, t_n) \Rightarrow \text{False})$

Exercise: FOL for GMP

- Write down logical representations for the following sentences, suitable for use with Generalized Modus Ponens:

• Horses, cows, and pigs are mammals	$Horse(x) \Rightarrow Mammal(x)$ $Cow(x) \Rightarrow Mammal(x)$ $Pig(x) \Rightarrow Mammal(x)$
• An offspring of a horse is a horse	$Offspring(x,y) \wedge Horse(y) \Rightarrow Horse(x)$
• Bluebeard is a horse	$Horse(Bluebeard)$
• Bluebeard is Charlie's parent	$Parent(Bluebeard, Charlie)$
• Offspring and parent are inverse relations	$Offspring(x, y) \Rightarrow Parent(y, x)$, $Parent(x, y) \Rightarrow Offspring(y, x)$
• Every mammal has a parent	$Mammal(x) \Rightarrow Parent(G(x), x)$, G Skolem function

Forward Chaining in FOL

- Forward Chaining (FC) consists of repeatedly applying GMP in all possible ways, adding to the initial KB all newly derived atomic sentences until no new sentence can be derived
- FC is normally triggered by the addition of new sentences into the KB, to derive all their consequences
- For instance, it can be used in the Wumpus game when new percepts are added to the KB, after each agent's move

Forward Chaining in FOL

- A simple (but inefficient) implementation of FC

```
function Forward-chaining (KB)
local variable: new
repeat
    new ← ∅ (the empty set)
    for each sentence s = (p1 ∧...∧ pn ⇒q)in KB do
        for each θ such that Subst(θ, p1 ∧ . . . ∧ pn) =
            Subst(θ,p1' ∧...∧ pn')for some p1',...,pn' ∈ KB do
            q' ← Subst(θ, q)
            if q' ∉ KB and q' ∉ new then add q' to new
    add new to KB
until new is empty
return KB
```

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*
  - We aim to derive Criminal(West)
- We need to represent these facts as FOL Horn clauses
  - . . . 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."
    - The sentence $\exists x\ Owns(Nono,x) \wedge Missile(x)$ is transformed into two definite clauses by Existential Instantiation, introducing a new constant M
    - $Owns(Nono,M)$
    - $Missile(M)$
  - "All of its missiles were sold to it by Colonel West"
    - $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Example

- We will also need to know that missiles are weapons:
  - Missile(x) ⇒Weapon(x)
- and we must know that 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)

Forward Chaining Example

- Summarizing (the universal quantifiers are not shown to keep the notation uncluttered)

$$
\begin{aligned}
&(American(x) \wedge Hostile(y) \wedge \\
&\quad Weapon(z) \wedge Sells(x, y, z)) \Rightarrow Criminal(x) && (1)\\
&(Missile(x) \wedge Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) && (2)\\
&Enemy(x, America) \Rightarrow Hostile(x) && (3)\\
&Missile(x) \Rightarrow Weapon(x) && (4)\\
&American(West) && (5)\\
&Enemy(Nono, America) && (6)\\
&Owns(Nono, M) && (7)\\
&Missile(M) && (8)
\end{aligned}
$$

Forward Chaining Example

- The FC algorithm carries out two repeat-until loops on the above KB
- No new sentences can be derived after the second loop
- First iteration
  - GMP to (2), (7) and (8), with $\{x/M\}$:
    (9) $Sells(West, Nono, M)$
  - GMP to (3) and (6), with $\{x/Nono, y/America\}$:
    (10) $Hostile(Nono)$
  - GMP to (4) and (8), with $\{x/M\}$:
    (11) $Weapon(M)$
- Second iteration
  - GMP to (1), (5), (10), (11) and (9), with $\{x/West, y/Nono, z/M\}$:
    (12) $Criminal(West)$

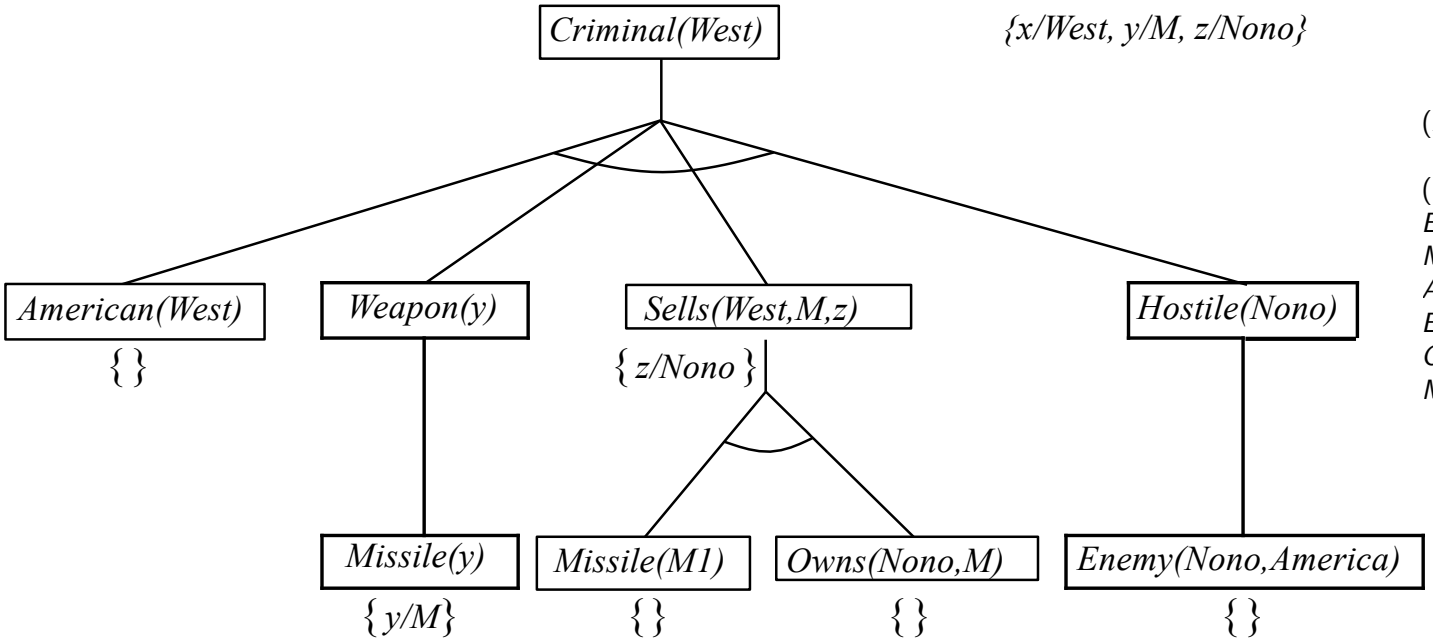
$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining in FOL

- The first-order Backward Chaining (BC) algorithm starts from a sentence (query) to be proven and recursively applies GMP backward
- Note that every substitution that is made to unify an atomic sentence with the consequent of an implication must be propagated back to every antecedent
- If the consequent of an implication unifies with more than one atomic sentence, at least one unification must allow the consequent to be proven

Backward Chaining Example

- A proof by BC can be represented as an And-Or graph
- The following graph (which should be read depth-first, left to right) shows the proof of the query *Criminal(West)* using the previous sentences (1)–(8) as the KB



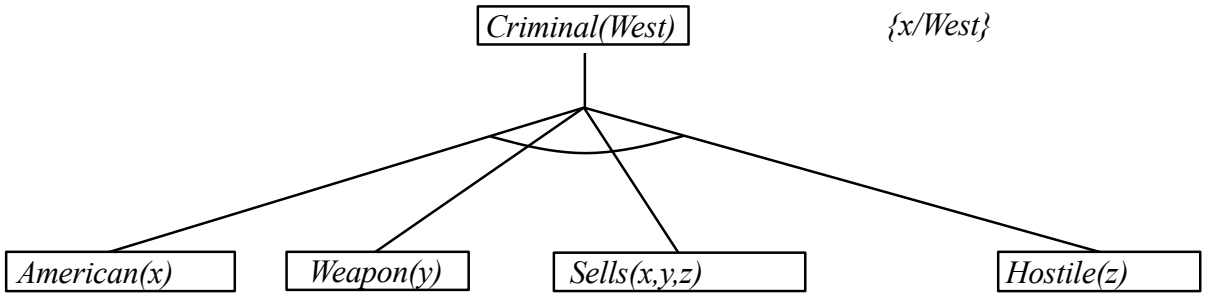
$(American(x) \wedge Hostile(y) \wedge$
$\quad Weapon(z) \wedge Sells(x, y, z)) \Rightarrow Criminal(x)$ (1)
$(Missile(x) \wedge Owns(Nono, x)) \Rightarrow Sells(West, Nono, x)$ (2)
$Enemy(x, America) \Rightarrow Hostile(x)$ (3)
$Missile(x) \Rightarrow Weapon(x)$ (4)
$American(West)$ (5)
$Enemy(Nono, America)$ (6)
$Owns(Nono, M)$ (7)
$Missile(M)$ (8)

Backward Chaining Example

$\boxed{Criminal(West)}$

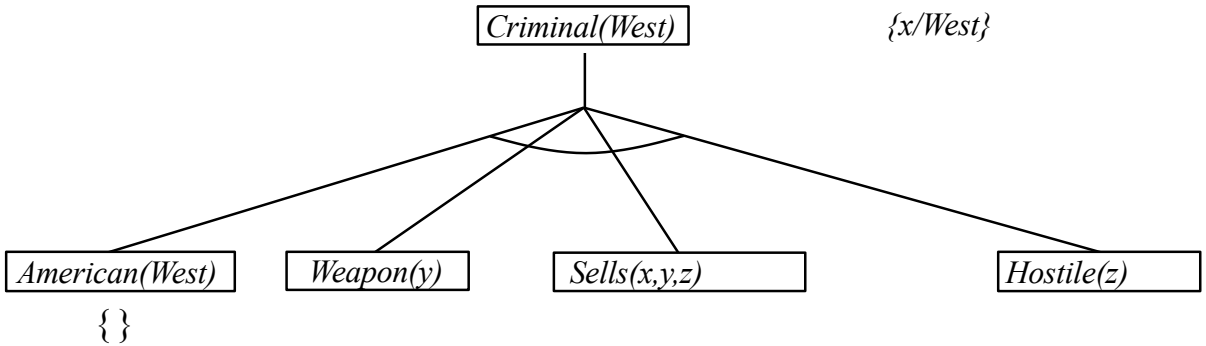
$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining Example



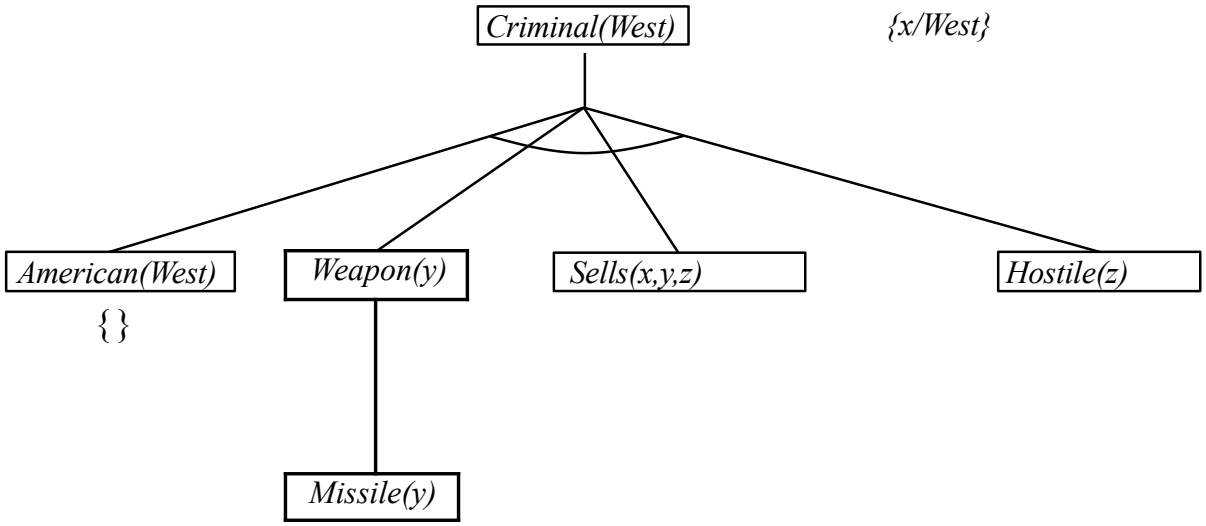
$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining Example



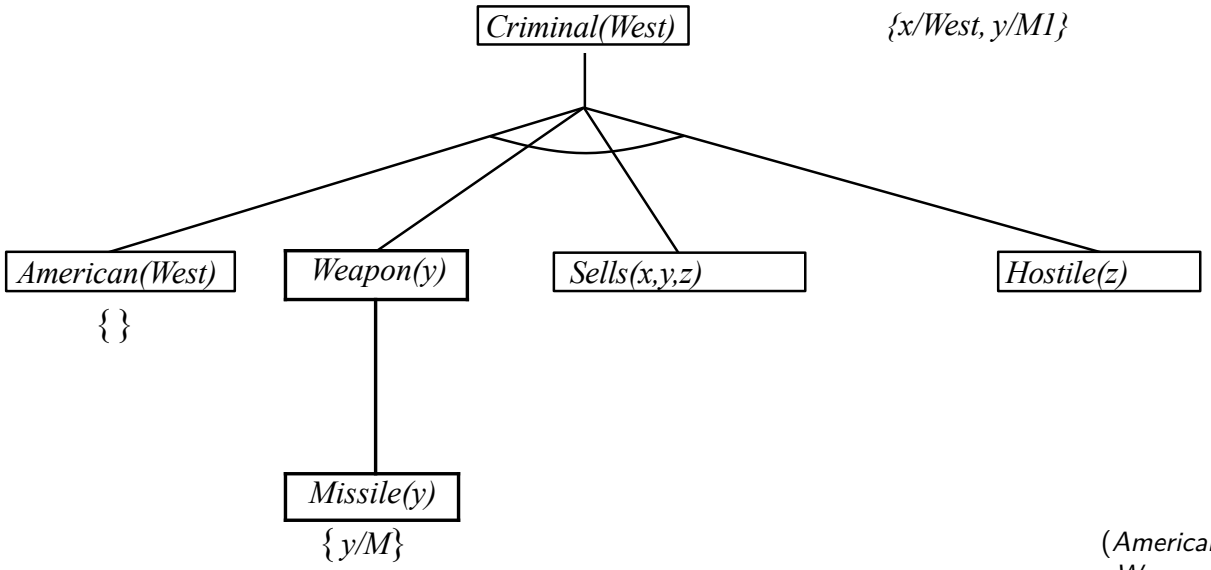
$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining Example



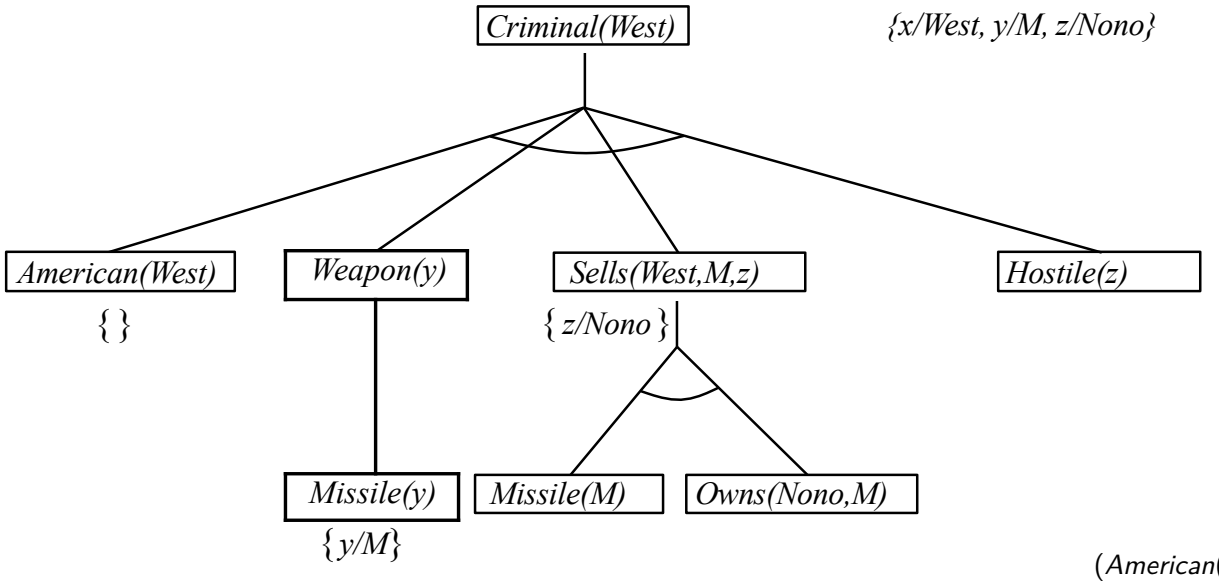
$$(American(x) \wedge Hostile(y) \wedge Weapon(z) \wedge Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \wedge Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining Example



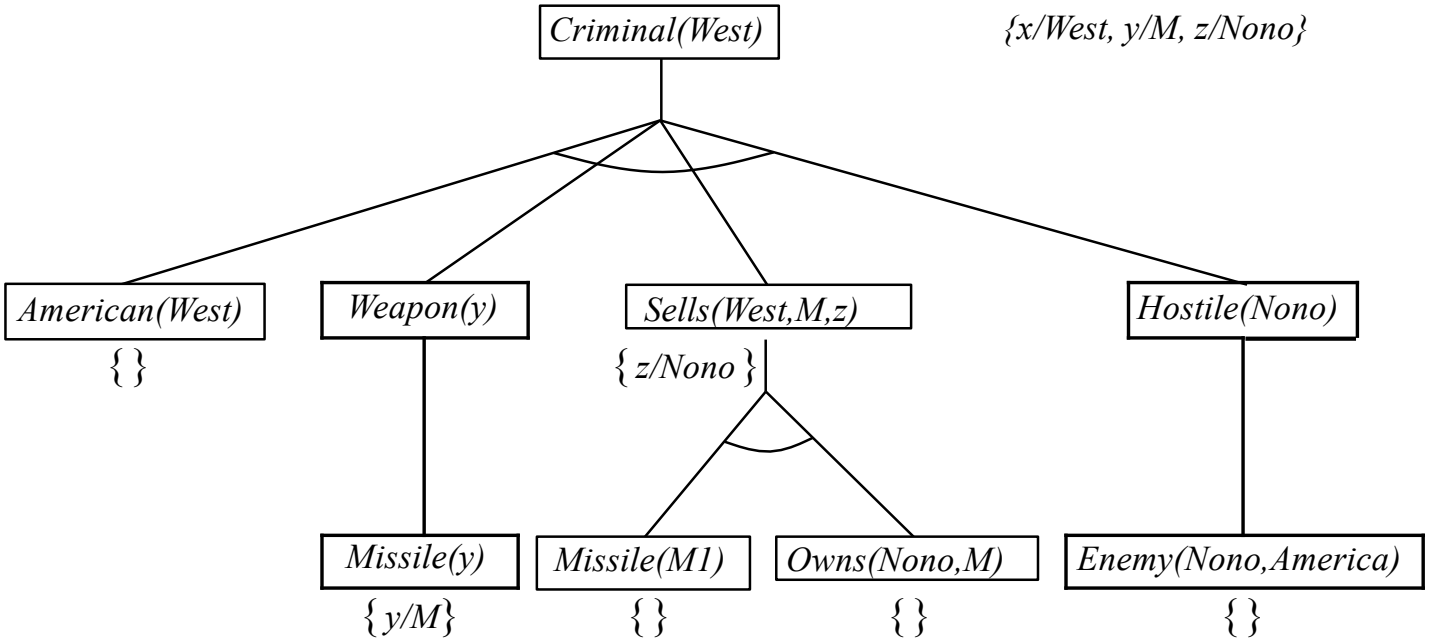
$(American(x) \wedge Hostile(y) \wedge$
$\quad Weapon(z) \wedge Sells(x, y, z)) \Rightarrow Criminal(x)$ (1)
$(Missile(x) \wedge Owns(Nono, x)) \Rightarrow Sells(West, Nono, x)$ (2)
$Enemy(x, America) \Rightarrow Hostile(x)$ (3)
$Missile(x) \Rightarrow Weapon(x)$ (4)
$American(West)$ (5)
$Enemy(Nono, America)$ (6)
$Owns(Nono, M)$ (7)
$Missile(M)$ (8)

Backward Chaining Example



$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$

Backward Chaining Example



$$(American(x) \land Hostile(y) \land Weapon(z) \land Sells(x, y, z)) \Rightarrow Criminal(x) \quad (1)$$
$$(Missile(x) \land Owns(Nono, x)) \Rightarrow Sells(West, Nono, x) \quad (2)$$
$$Enemy(x, America) \Rightarrow Hostile(x) \quad (3)$$
$$Missile(x) \Rightarrow Weapon(x) \quad (4)$$
$$American(West) \quad (5)$$
$$Enemy(Nono, America) \quad (6)$$
$$Owns(Nono, M) \quad (7)$$
$$Missile(M) \quad (8)$$