Week 11: First Order Logic

Dr Ammar Masood

Department of Cyber Security,

Air University Islamabad

Based on James D. Skrentny notes from https://pages.cs.wisc.edu/~skrentny/cs540/

Contents

- More on Representation
- Syntax and Semantics of First-Order Logic
- Using First Order Logic
- Knowledge Engineering in First-Order Logic
- Inference in FOL

A Brief History of Reasoning

• 450BCE	Stoics	PL, inference (?)
• 32BCE	Aristotle	inference rules (syllogisms), quantifiers
• 1565	Cardano	PL + uncertainty (probability theory)
• 1847	Boole	PL (again)
• 1879	Frege	FOL
• 1922	Wittgenstein	proof using truth table
• 1930	Gödel	complete algo for FOL *exists*
• 1930	Herbrand	complete algo for FOL (reduce to PL)
• 1931	Gödel	complete algo doesn't exist if induction used
• 1960	Davis/ Putnam	practical algo for PL
• 1965	Robinson	practical algo for FOL (resolution)

First-Order Logic

- AKA First-Order Predicate Logic
- AKA First-Order Predicate Calculus

- Much more powerful the propositional (Boolean) logic
  - Greater expressive power than propositional logic
    - We no longer need a separate rule for each square to say which other squares are breezy/pits
  - Allows for facts, objects, and relations
    - In programming terms, allows classes, functions and variables

Pros and Cons of Propositional Logic

- + Propositional logic is declarative: pieces of syntax correspond to facts
- + Propositional logic allows for partial / disjunctive / negated information (unlike most data structures and DB
- + Propositional logic is compositional: the meaning of B_{11} ^ P_{12} is derived from the meaning of B_{11} and P_{12}
- + Meaning of propositional logic is context independent: (unlike natural language, where the meaning depends on the context)

- - Propositional logic has very limited expressive power: (unlike natural language)
  - E.g. cannot say Pits cause Breezes in adjacent squares except by writing one sentence for each square

Pros of First-Order Logic

- First-Order Logic assumes that the world contains:
  - Objects
    - E.g. people, houses, numbers, theories, colors, football games, wars, centuries, …
  - Relations
    - E.g. red, round, prime, bogus, multistoried, brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, …
  - Functions
    - E.g. father of, best friend, third quarter of, one more than, beginning of, …

Logics in General

Language	**Ontological Commitment** (What exists in the world)	**Epistemological Commitment** (What an agent believes about facts)
Propositional Logic	Facts	True / False / Unknown
First-Order Logic	Fact, objects, relations	True / False / Unknown
Temporal Logic	Facts, objects, relations, times	True / False / Unknown
Probability Theory	Facts	Degree of belief $\in$ [0,1]
Fuzzy Logic	Degree of truth $\in$ [0,1]	Known interval value

FOL Syntax

- FOL fixes problems with PL:
  - PL doesn't have variables.
    FOL does.
  - Identifying individuals in PL is hard.
    FOL it's easy.
  - PL can't directly express properties of individuals or relations between individuals.
    FOL can.

- Inferencing in PL is fairly easy.
  In FOL it is more complicated.

Syntax of First-Order Logic

• Constants	KingJohn, 2, …
• Predicates	Brother, >, …
• Functions	Sqrt, LeftArmOf, …
• Variables	x, y, a, b, …
• Connectives	$\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
• Equality	=
• Quantifiers	$\exists$ $\forall$

FOL Syntax

- A term denotes an object in the world.
  - **Constant:** **`BobSmith`**, 2, **`Madison`**, **`Green`**, …
  - **Variable:** **`x`**, **`y`**, **`a`**, **`b`**, **`c`**, …
  - **Function(Term_1, …, Term_n):**
    `Sqrt(9)`, **`Distance(Madison,Milwaukee)`**
    - is a relation for which there is one answer
    - maps one or more objects to another *single object*
    - can refer to an unnamed object: e.g. **`LeftLegOf(John)`**
    - represents a user defined *functional* relation
    - cannot be used with logical connectives
- A ground term is a term with no variables.

FOL Syntax

- An atom/literal is smallest expression to which a truth value can be assigned.
  - **Predicate($Term_1$, …, $Term_n$):**
    `Teacher(John, Deb)`, `<=(Sqrt(2),Sqrt(7))`
    - is a relation for which there is more than one answer
    - maps one or more objects to a *truth value*
    - represents a user defined *truth* relation
  - **$Term_1$ = $Term_2$:**
    `Income(John) = 20K`, `1 = 2`
    - represents the *equality* relation when two terms refer to the same object
    - is a predicate in prefix form: =($Term_1$, $Term_2$)

FOL Syntax

- A sentence represents a fact in the world that is assigned a truth value.
  - atom
  - complex sentence using connectives: $\wedge \vee \neg \Rightarrow \Leftrightarrow$

    Brother (John, Richard) $\wedge \neg$Brother (John, Father(John))

    >(11,22) $\wedge$ <(22,33)
  - complex sentence using quantified variables: $\forall \exists$

    more in a bit…

FOL Syntax: Assigning Truth

✱ *Sentences are assigned a truth value with respect to a model and an interpretation.*

- The model contains the objects and the relations among them.
  - the **domain** of a model is the set of objects it contains

- The interpretation specifies what the symbols refer to:
  - constants symbols refer to objects
  - predicate symbols refer to truth relations
  - functional symbols refer to functional relations

FOL Semantics: Assigning Truth

✱ *The atom Predicate(Term$_1$, …, Term$_n$) is true iff the objects referred to by Term$_1$, …, Term$_n$ are in the relation referred to by the predicate.*

- What is the truth value for `F(D,J)`?
  - model

    objects: **`Deb, Jim, Sue, Bob`**

    relation: **`Friend {<Deb,Sue>, <Sue,Deb>}`**
  - interpretation

    `D` means Deb, **`J`** means Jim, **`S`** means Sue, **`B`** means Bob

    F(Term$_1$,Term$_2$) means Term$_1$ is friend of Term$_2$

FOL Syntax: Quantifiers

Universal quantifier: $\forall$<variables> <sentence>

✱ *Means the sentence holds true*
for all values of x in the domain of variable x.

- Main connective typically $\Rightarrow$ forming if-then rules
  - ***All** humans are mammals.*
    $\forall$**x Human(x)** $\Rightarrow$ **Mammal(x)**
    for all x if x is a human then x is a mammal
  - ***Mammals must have fur.***
    $\forall$**x Mammal(x)** $\Rightarrow$ **HasFur(x)**
    for all x if x is a mammal then x has fur

FOL Syntax: Quantifiers

$\forall$`x Human(x)` $\Rightarrow$ `Mammal(x)`

- Equivalent to conjunction of instantiations of x:

```
(Human(Jim) ⇒ Mammal(Jim)) ∧
(Human(Deb) ⇒ Mammal(Deb)) ∧
(Human(22)  ⇒ Mammal(22)  ) ∧ …
```

- Common mistake is to use $\wedge$ as main connective.
  - results in a blanket statement about everything
- Bad example $\forall$`x Human(x)` $\wedge$ `Mammal(x)` means?
  - everything is human and a mammal

```
(Human(Jim) ∧ Mammal(Jim)) ∧
(Human(Deb) ∧ Mammal(Deb)) ∧
(Human(22)  ∧ Mammal(22)  ) ∧ …
```

FOL Syntax: Quantifiers

Existential quantifier: ∃<variables> <sentence>

✱ *Means the sentence holds true*
for some value of x in the domain of variable x.

- Main connective typically ∧
  - *Some humans are old.* **Becomes what in FOL?**
    ∃x Human(x) ∧ Old(x)
    there exist an x such that x is a human and x is old
  - *Mammals may have arms.* **Becomes what in FOL?**
    ∃x Mammal(x) ∧ HasArms(x)
    there exist an x such that x is a mammal and x has arms

FOL Syntax: Quantifiers

$\exists$`x Human(x)` $\wedge$ `Old(x)`

- Equivalent to disjunction of instantiations of x:

```
(Human(Jim) ∧ Old(Jim)) ∨
(Human(Deb) ∧ Old(Deb)) ∨
(Human(22)  ∧ Old(22) ) ∨ …
```

- Common mistake is to use $\Rightarrow$ as main connective.
  - results in a weak statement

FOL Syntax: Quantifiers

- $\forall$x $\forall$y Likes(x,y) is what in English?
  Everyone likes everyone. It's the *active voice.*
- $\forall$y $\forall$x Likes(x,y) is what in English?
  Everyone is liked by everyone. It's the *passive voice.*

☞ Do these mean the same thing?

- Property of quantifiers:
  - $\forall \mathbf{x}\, \forall \mathbf{y}$ is the same as $\forall \mathbf{y}\, \forall \mathbf{x}$
  - $\exists \mathbf{x}\, \exists \mathbf{y}$ is the same as $\exists \mathbf{y}\, \exists \mathbf{x}$
  - note: $\exists \mathbf{x}\, \exists \mathbf{y}$ can be written as $\exists \mathbf{x},\mathbf{y}$, likewise with $\forall$

FOL Syntax: Quantifiers

- $\forall x\, \exists y$ `Likes(x,y)` is what in English?
  Everyone likes someone. *again the active voice*
- $\exists y\, \forall x$ `Likes(x,y)` is what in English?
  Someone is liked by everyone. *again the passive voice*

☞ Do these mean the same thing?

- Property of quantifiers:
  - $\forall \mathbf{x}\, \exists \mathbf{y}$ is not the same as $\exists \mathbf{y}\, \forall \mathbf{x}$
  - $\exists \mathbf{x}\, \forall \mathbf{y}$ is not the same as $\forall \mathbf{y}\, \exists \mathbf{x}$

FOL Syntax: Quantifiers

- $\forall$x Likes(x,IceCream) is what in English?
  Everyone likes ice cream.
- $\neg\exists$x $\neg$Likes(x,IceCream) is what in English?
  There is No one who doesn't like ice cream. It's a double negative!

☞ Do these mean the same thing?

- Properties of quantifiers:
  - **$\forall$x P(x)** is the same as **$\neg\exists$x $\neg$P(x)**
  - **$\exists$x P(x)** is the same as **$\neg\forall$x $\neg$P(x)**
  - This is the negation of the application of de Morgan's law to the fully instantiated sentence.

FOL Syntax: Quantifiers

- $\forall$x Likes(x,IceCream) is what in English?
  Everyone likes ice cream.

☞ What is the logical negation of this sentence?
Not everyone likes ice cream. which is the same as…

$\exists$x $\neg$Likes(x,IceCream) is what in English?
Someone doesn't like ice cream.

- 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)**
  - This is from the application of de Morgan's law to the fully instantiated sentence.

FOL Syntax: Basics

- A free variable is a variable that isn't bound by a quantifier.
  - `∃y Likes(x,y)`   `x` is free, `y` is bound

- A well-formed formula is a sentence where all variables are quantified.

Fun with Sentences

- One's mother is one's female parent

  $\forall x,y \;\; Mother(x,y) \Leftrightarrow (Female(x) \wedge Parent(x,y))$

- A first cousin is a child of a parent's sibling

  $\forall x,y \; FirstCousin(x,y) \Leftrightarrow \exists p,ps \; Parent(p,x) \wedge Sibling(ps,p) \wedge (Parent(ps,y)$

Thinking in FOL

☞ Convert the following English sentences to FOL.

- *Bob is a fish.*
  - What are the objects?
    Bob look for nouns and noun phrases
  - What are the relations?
    is a fish look for verbs and verb phrases

  Answer: `Fish(Bob)` a unary relation or property

- *Deb and Sue are women.* we'll be casual about plurals
- *Deb and Sue aren't plants.* ambiguous?
- *Deb and Sue aren't friends.* use a function? predicate?

Thinking in FOL

☞ Convert the following English sentences to FOL.

- *America bought Alaska from Russia.*
  - What are the objects?
    America, Alaska, Russia
  - What are the relations?
    bought(who, what, from) an **n-ary relation** where n is 3

  Answer: `Bought(America,Alaska,Russia)`

✱ *The model must include the ordering and meaning of a predicate's terms.*

- *Warm is between cold and hot.*
- *Deb, Lynn, Jim, and Steve went together to APT.*

Thinking in FOL

☞ Now lets think about quantifying variables.

- *Jim collects everything.*
  - What are the objects?
    Jim
  - What are the variables and how are they quantified?
    everything **x**, all - universal

  Answer: ∀x Collects(Jim,x)

  Collects(Jim,Pencil) ∧ **Collects(Jim,Deb)** ∧ …

- *Jim collects something.*
- *Somebody collects Jim.* How do you handle "body"?

Thinking in FOL

When to restrict the domain, e.g. people:

- All: $\forall x\ Person(x) \wedge ... \Rightarrow ...$
  - **things**: anything, everything, whatever
  - **people**: anybody,anyone,everybody,everyone,whoever
- Some (at least one): $\exists x\ Person(x) \wedge ... \wedge ...$
  - **things**: something
  - **people**: somebody, someone
- None: $\neg\exists x\ Person(x) \wedge ... \wedge ...$
  - **things**: nothing
  - **people**: nobody, no one

Thinking in FOL

☞ How about sentences with multiple variables?

- *Somebody collects something.*
  - What are the objects?
    none!
  - What are the variables and how are they quantified?
    somebody **x** and something **y**, some - existential

  Answer: $\exists$ x,y Person(x) $\wedge$ Collects(x,y)

- *Everybody collects everything.*
- *Everybody collects something.*
- *Something is collected by everybody.*

Thinking in FOL

☞ Convert the following English sentences to FOL.

- *Nothing collects anything.*
  - What are the variables?
    nothing **x** and anything **y**
  - How are they quantified?
    not one (i.e. not existential) and all (universal)

  Answer: `¬∃x ∀y Collects(x,y)`

☞ What's the "double-negative" equivalent?

Everything does not collect anything.

Answer: `∀x,y ¬Collects(x,y)`

- *Everything collects nothing.*

Thinking in FOL

☞ Complex quantified sentences:

- *Any good amateur can beat some professional.*
  - **break into components**
  - $\forall$**x** [(x is a good amateur) $\Rightarrow$
    (x can beat some professional)]
  - (x can beat some professional) **becomes**
    $\exists$**y** [(y is a professional) $\wedge$ (x can beat y)]

Answer: $\forall$**x [(Amateur(x) $\wedge$ GoodPlayer(x)) $\Rightarrow$**
$\exists$y (Professional(y) $\wedge$ Beat(x,y))]

Thinking in FOL

- Interesting words: *always*, *sometimes*, *never*
  - *Good people always have friends.*
    could mean: *All good people have friends.*
    $\forall x\ Person(x) \land Good(x) \Rightarrow \exists y(Friend(x,y))$
  - *Busy people sometimes have friends.*
    could mean: *Some busy people have friends.*
    $\exists x\ Person(x) \land Busy(x) \land \exists y(Friend(x,y))$
  - *Bad people never have friends.*
    could mean: *Bad people have no friends.*
    $\forall x\ Person(x) \land Bad(x) \Rightarrow \neg\exists y(Friend(x,y))$
    or equivalently: *No bad people have friends.*
    $\neg\exists x\ Person(x) \land Bad(x) \land \exists y(Friend(x,y))$

Equality

- We allow the usual infix = operator
  - Father(John) = Henry
  - $\forall x,\ sibling(x, y) \Rightarrow \neg(x=y)$

- Generally, we also allow mathematical operations when needed, e.g.
  - $\forall x,y,\ NatNum(x) \wedge NatNum(y) \wedge x = (y+1) \Rightarrow x > y$

- Example: (Sibling in terms of Parent)

  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x=y) \wedge \exists m,f\ \neg(m=f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

Summary of FOL Representation

- **Constants**: `Bob, 2, Madison, ...`
- **Functions**: `Income, Address, Sqrt, ...`
- **Predicates**: `Sister, Teacher, <=, ...`
- **Variables**: `x, y, a, b, c, ...`
- **Connectives**: $\neg \wedge \vee \Rightarrow \Leftrightarrow$
- **Equality**: $=$
- **Quantifiers**: $\forall \exists$

Propositional Logic vs FOL

B33 → (P32 v P 23 v P34 v P 43) , similar for all internal squares…

"Internal squares adjacent to pits are breezy":

$\forall$ X,Y (B(X,Y) $\wedge$ (X > 1) $\wedge$ (Y > 1) $\wedge$ (Y < 4) $\wedge$ (X < 4)) ←→

(P(X-1,Y) v P(X,Y-1) v P(X+1,Y) v (X,Y+1))

Interacting with FOL KBs

- Tell the system assertions
  - Facts :
    - Tell (KB, person (John))
  - Rules:
    - Tell (KB, $\forall$x, person(x) $\Rightarrow$ likes(x, McDonalds))
- Ask questions
  - Ask (KB, person(John))
  - Ask (KB, likes(John, McDonalds))
  - Ask (KB, likes(x, McDonalds))

Types of Answers

- Fact is in the KB
  - Yes.
- Fact is not in the KB
  - Yes (if it can be proven from the KB)
  - No (otherwise)

Interacting with FOL KBs

- Suppose a wumpus-world agent is using a FOL KB and perceive a smell and breeze (but no glitter) at t=5

- TELL(KB, Percept([Smell, Breeze, None],5))
- ASKVARS(KB, ∃a Action(a, 5))
  - i.e. does the KB entail any particular action at t=5?

- Answer: Yes, {a/Shoot}   <- substitution (binding list)

Knowledge Base for Wumpus World

- "Perception"
  - $\forall t, s,g,w,c$ Percept([s,Breeze,g,w,c], t) $\Rightarrow$ Breeze(t)
  - $\forall t, s,g,w,c$ Percept([s,None,g,w,c], t) $\Rightarrow \neg$Breeze(t)
  - $\forall t, s,b,w,c$ Percept([s,b,Glitter,w,c], t) $\Rightarrow$ Glitter(t)
  - $\forall t, s,b,w,c$ Percept([s,b,None,w,c], t) $\Rightarrow \neg$Glitter(t)
- "Reflex"
  - $\forall t$ Glitter(t) $\Rightarrow$ BestAction(Grab, t)
- "Reflex with internal state"
  - $\forall$ t AtGold(t) $\wedge$ ¬Holding(Gold, t) $\Rightarrow$ Action(Grab, t)

- Holding(Gold, t) cannot be observed
  - Keeping track of change is essential!!!!

Deducing Hidden Properties

- Properties of locations:
  $\forall x, t\ At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(x)$
  $\forall x, t\ At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(x)$

- Squares are breezy near a pit:
  - Diagnostic Rule – infer cause from effect
    - $\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \wedge Adjacent(x, y)$
  - Causal Rule – infer effect from cause
    - $\forall x,y\ Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(x,y)$

Deducing Hidden Properties

- **Definition for the Breezy predicate:**
  - If a square is breezy, some adjacent square must contain a pit
    - $\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \wedge Adjacent(x, y)$
  - If a square is not breezy, no adjacent pit contains a pit
    - $\forall y\ \neg Breezy(y) \Rightarrow \neg\exists x\ Pit(x) \wedge Adjacent(x, y)$
  - **Combining these two…**
    - $\forall y\ \mathbf{Breezy(y)} \Leftrightarrow \exists x\ \mathbf{Pit(x)} \wedge \mathbf{Adjacent(x, y)}$

Knowledge engineering in FOL

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

Inference Rules for FOL

- Universal Elimination, UE
  variable substituted with ground term

  $\forall$`x Eats(Jim,x)` infer `Eats(Jim,Cake)`

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

- **Existential Elimination, EE**
  variable substituted with **new** constant called a Skolem constant

  $\exists$**`x Eats(Jim,x)`** infer **`Eats(Jim,K)`**

$$\frac{\exists v\ \alpha}{\mathrm{SUBST}(\{v/K\},\ \alpha)}$$

- **Using these two inference rules on a FOL knowledge base enables it to be propositionalized, i.e., variables can all be eliminated.**
- **Then natural deduction can be done using inference rules for PL.**

PL Inference Rules also for FOL

- Implication Elimination (IE) (Modus Ponens, MP) $$\frac{\alpha \Rightarrow \beta,\ \ \alpha}{\beta}$$
- **And Elimination** (AE) $$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$
- **And Introduction** (AI) $$\frac{\alpha_1,\ \alpha_2,\ \ldots\ ,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$
- **Or Introduction** (OI) $$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$
- **Double-Negation Elimination** (DNE) $$\frac{\neg\,\neg\alpha}{\alpha}$$
- **DeMorgan's Rule** (D) and likewise for $\neg\,(\alpha \wedge \beta)$ $$\frac{\neg\,(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta)}$$

PL Inference Rules also for FOL

- Unit Resolution (UR)

$$\frac{\alpha \vee \beta, \neg \beta}{\alpha}$$

- **Resolution** (R)

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

equivalently
(transitivity of implication)

$$\frac{\neg\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

A Simple FOL Proof using Natural Deduction

- *Jim is a turtle.*
  1. **Turtle(Jim)**

- *Deb is a rabbit.*
  2. **Rabbit(Deb)**

- *Turtles outlast Rabbits.*
  3. $\forall$**x,y Turtle(x)** $\wedge$ **Rabbit(y)** $\Rightarrow$ **Outlast(x,y)**

- Query: *Jim outlasts Deb.*
  - **Outlast(Jim,Deb)**

A Simple FOL Proof using Natural Deduction

- And Introduction AI 1. & 2.

  4. `Turtle(Jim)` $\wedge$ `Rabbit(Deb)`

- Universal Elimination UE 3. {x/Jim, y/Deb}

  5. `Turtle(Jim)` $\wedge$ `Rabbit(Deb)` $\Rightarrow$ `Outlast(Jim,Deb)`

- Modus Ponens MP 4. & 5.

  6. `Outlast(Jim,Deb)`

- *Jim is a turtle.*
  1. `Turtle(Jim)`
- *Deb is a rabbit.*
  2. `Rabbit(Deb)`
- *Turtles outlast Rabbits.*
  3. $\forall$`x,y Turtle(x)` $\wedge$ `Rabbit(y)` $\Rightarrow$ `Outlast(x,y)`
- Query: *Jim outlasts Deb.*
  - `Outlast(Jim,Deb)`

✱ *AI, UE, MP is a common inference pattern.*

✱ *Automated inference harder with FOL than PL.*
Variables can take on a potentially infinite number of possible values from their domain and thus UE can be applied in a potentially infinite number of ways to KB.

Generalized Modus Ponens (GMP)

* *Unify rule premises with known facts and apply unifier to conclusion.*

- Rule:

  $\forall x,y\ Turtle(x) \wedge Rabbit(y) \Rightarrow Outlast(x,y)$

  Known facts: Turtle(Jim), Rabbit(Deb)

  Unifier: {x/Jim, y/Deb}

- Apply unifier to conclusion: Outlast(Jim,Deb)

Generalized Modus Ponens (GMP)

- Combines AI, UE, and MP (IE) 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

$SUBST(\theta, \alpha)$ **means apply substitutions in θ to α**

- **Substitution list** $\theta = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$ **means**
  - replace all occurrences of variable v_i with term t_i
  - substitutions are made in left to right order

Generalized Modus Ponens (GMP)

- Combines AI, UE, and MP (IE) 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

- **All variables *assumed* to be universally quantified**

- **Used with a KB in Horn normal form:**

definite clause: disjunction of literals with exactly 1 positive literal
  - **fact**: single positive literal $P_1(x)$, $P_2(x)$
  - **rule**: conjunction of atoms $\Rightarrow$ atom $P_1(x) \wedge P_2(x) \Rightarrow Q(x)$
    has only one positive literal $\neg P_1(x) \vee \neg P_2(x) \vee Q(x)$

Generalized Modus Ponens (GMP)

- Combines AI, UE, and MP (IE) 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

For Example:

p_1' = `Smarter(Deb,Bob)`

p_2' = `Smarter(Bob,Joe)`

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

☞ **What substitutions are needed?**

θ = $\{x/Deb, y/Bob, z/Joe\}$

$SUBST(\theta, q)$ = `Smarter(Deb,Joe)`

Unification

✱ *Substitution θ unifies p_1' and p_1 if $SUBST(\theta, p_1') = SUBST(\theta, p_1)$.*

p_1'	p_1	θ
`Turtle(y)`	`Turtle(Jim)`	$\{y/Jim\}$
`Hears(Deb,x)`	`Hears(Deb,Sue)`	$\{x/Sue\}$
`Hears(Deb,x)`	Hears(~~x~~y,Jim)	$\{y/Deb, x/Jim\}$

✱ **Variables must be standardized apart!**

If the same variable(s) is found in both p_1' and p_1 then rename variable(s) so none are shared.

Unification take two sentences and return a unifier for them (a substitution) if it exists

Unification

- *Substitution* θ *unifies* p_1' *and* p_1
  if $SUBST(\theta, p_1') = SUBST(\theta, p_1)$.

p_1'	p_1	θ
`Turtle(y)`	`Turtle(Jim)`	$\{y/Jim\}$
`Hears(Deb,x)`	`Hears(Deb,Sue)`	$\{x/Sue\}$
`Hears(Deb,x)`	`Hears(y,Jim)`	$\{y/Deb, x/Jim\}$
`Hears(Deb,x)`	`Hears(z,Mother(z))`	$\{z/Deb, x/Mother(Deb)\}$
`Eats(y,y)`	`Eats(z,Fish)`	$\{y/z, z/Fish\}$
`Sees(Jo,x,y)`	`Sees(z,Jim,At(z))`	$\{z/Jo, x/Jim, y/At(Jo)\}$
`Sees(x, ID(x),At(Jo))`	`Sees(Jim, ID(y),At(y))`	*Failure since* $At(Jo) \neq At(Jim)$

Unification: Simplified Algorithm

```
//see figure 9.1 of text for full implementation
//returns a unifier or null if failure
//assumes predicates match and variables are separated apart
List unify (Literal m, Literal n, List theta) {
  scan m and n left-to-right and find the first corresponding terms where m and
  n are not the same
  if (no difference in terms) return theta; //success
  else {r = term in m; s = term in n;} //where term r != term s
  if (isVariable(r)) {
    theta = unionOf(theta,{r/s});//should fail if r occurs in s
    unify(substitute(theta, m), substitute(theta, n), theta);
  }
  else if (isVariable(s)) {
    theta = unionOf(theta,{s/r});//should fail if s occurs in r
    unify(substitute(theta, m), substitute(theta, n), theta);
  }
  else return null; //failure
}
```

Completeness of FOL Automated Inference

- Truth table enumeration: incomplete for FOL
  table may be infinite in size for infinite domain
- Natural Deduction: complete for FOL but impractical since branching factor too large
- GMP: incomplete for FOL
  not every sentence can be converted to Horn form
- GMP: complete for FOL KB in HNF
  - **forward chaining:** move from KB to query
  - **backward chaining:** move from query to KB

Forward Chaining (FC) with GMP

✱ *Move "forwards" from KB to query*

- Simplified FC Algorithm (see figure 9.3):
  Assume query *q* is asked of KB

```
repeat until no new sentences are inferred
    initialize NEW to empty
    for each rule that can have all of its premises satisfied
        apply composed substitution to the conclusion
        add the new conclusion to NEW if it's not just a renaming
        done if the new conclusion unifies with the query
    add sentences in NEW to KB
return false since q never concluded
```

FC: Example Knowledge Base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy America, has some missiles, and all of its missiles were sold to it by Col. West, who is an American.

- Prove that Col. West is a criminal.

FC: Example Knowledge Base

- …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
  $\exists x\ \mathrm{Owns}(\mathrm{Nono}, x) \wedge \mathrm{Missiles}(x)$

  $Owns(Nono, M_1)$ *and* $Missle(M_1)$
- …all of its missiles were sold to it by Col. West
  $\forall x\ Missle(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
- Missiles are weapons
  $Missle(x) \Rightarrow Weapon(x)$

FC: Example Knowledge Base

- An enemy of America counts as “hostile”
  Enemy(x, America) $\Rightarrow$ *Hostile(x)*

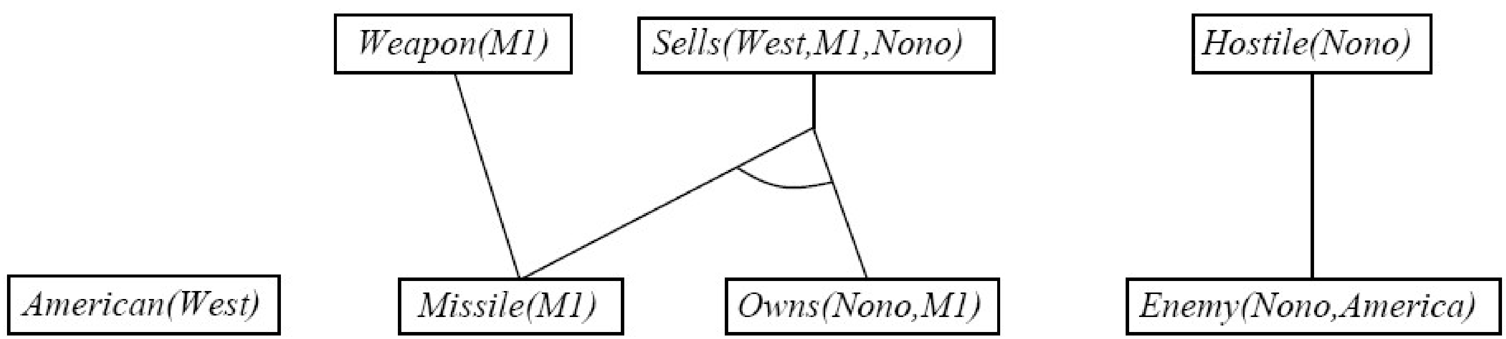
- Col. West who is an American
  American(Col. West)

- The country Nono, an enemy of America
  Enemy(Nono, America)

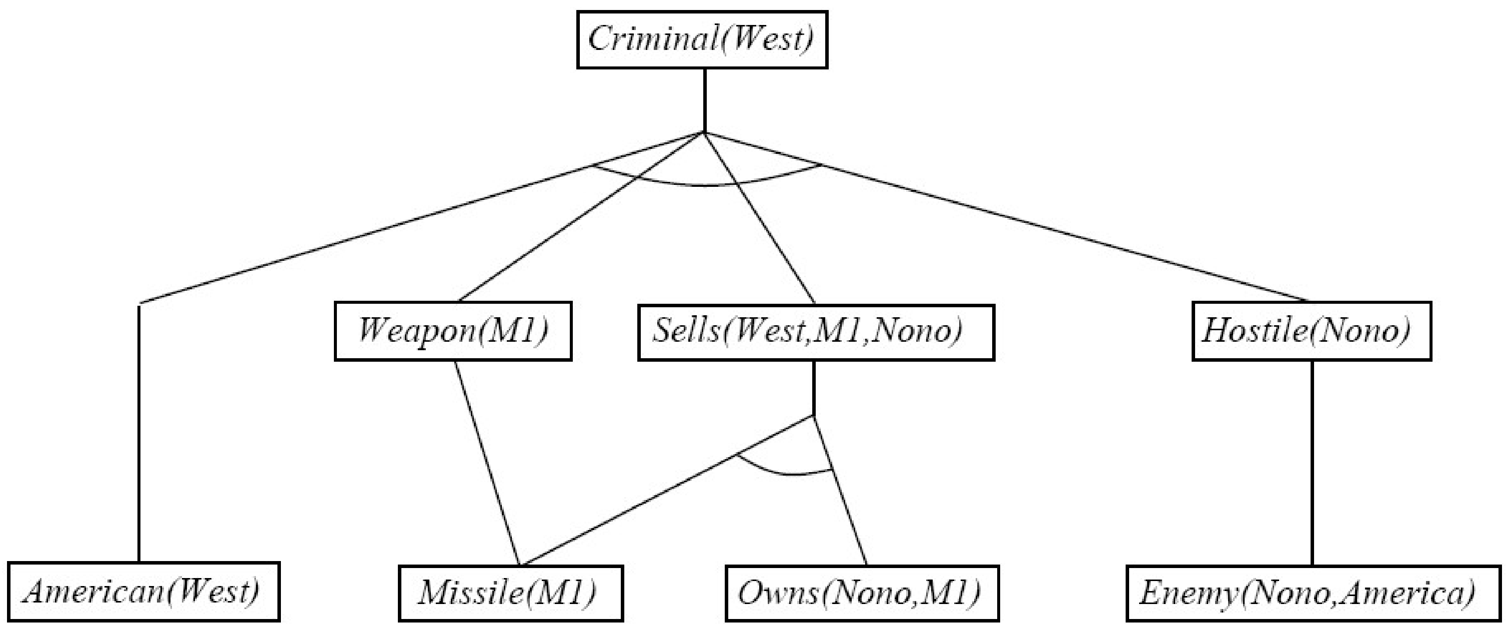
FC: Example Knowledge Base

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

FC: Example Knowledge Base



FC: Example Knowledge Base



Limits of GMP

- FC and BC are complete for Horn KBs
  but are incomplete for general FOL KBs:

```
PhD(x)   ⇒ HighlyQualified(x)
¬PhD(x)  ⇒ EarlyEarnings(x)
HighlyQualified(x) ⇒ Rich(x)
EarlyEarnings(x)   ⇒ Rich(x)
Query: Rich(Me)
```

☞ What is the problem with the example above?

- Is there a complete inferencing algorithm
  that works for any FOL knowledge base?

  Yes! Resolution Refutation

Resolution Refutation

- Resolution refutation is
  an inferencing technique
  that requires a CNF representation.

✱<u>Any</u> *FOL KB can be converted into CNF.*

- CNF: Conjunctive Normal Form

  conjunction of clauses where
  - **CNF clause: a disjunction of literals**

    e.g. `Hot(x)` ∨ `Warn(x)` ∨ `Cold(x)`
  - **Literal: an atom**

    either positive (unnegated) or negative (negated)

    e.g. ¬`Happy(Sally), Rich(x)`

Resolution Refutation: The Inference Rule

- **Resolution refutation uses the resolution rule generalized for FOL:**

PL Resolution Rule (R)

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

Generalized Resolution Rule (GR) in FOL:

where l_i and m_i are literals for all i
where $UNIFY(l_j, m_k) = \theta$, and m_k is the negation of l_j

$$\frac{l_1 \vee \ldots\ l_j \vee \ldots \vee l_m,\quad m_1 \vee \ldots\ m_k \vee \ldots \vee m_n}{SUBST(\theta, l_1 \vee \ldots\ l_{j-1} \vee l_{j+1} \ldots \vee l_m \vee m_1 \vee \ldots\ m_{k-1} \vee m_{k+1} \ldots \vee m_n)}$$

Resolution Refutation: GMP Example

$$\frac{\texttt{Fulfilled(Me)},\ \neg\texttt{Fulfilled(x)} \vee \texttt{Happy(x)}}{SUBST(\theta, \texttt{Happy(x)})}$$

- l_j is **`Fulfilled(Me)`**
  m_k is $\neg$**`Fulfilled(x)`**

- $UNIFY(l_j, m_k)$ results in $\theta = \{x/Me\}$
  $SUBST(\theta,$ **`Happy(x)`**$)$ results in ?

  Inferred sentence: **`Happy(Me)`**

∗ ***GMP is special case of generalized resolution.***

Summary

- FOL is a more expressive language
  but inferencing is more complicated

- Inferencing in FOL can be done by:
  - **Propositionalizing** the FOL KB
    (universal and existential elimination rules)
    and using sound inference rules from PL
    - complete for FOL but impractical
  - Converting KB to **Horn Normal Form**
    and using **Generalized Modus Ponens** rule
    - complete for HNF, but not all FOL KBs can be converted to HNF
  - Converting KB to **Conjunctive Normal Form**
    and using **Resolution Refutation**
    - complete, all FOL KB can be converted to CNF