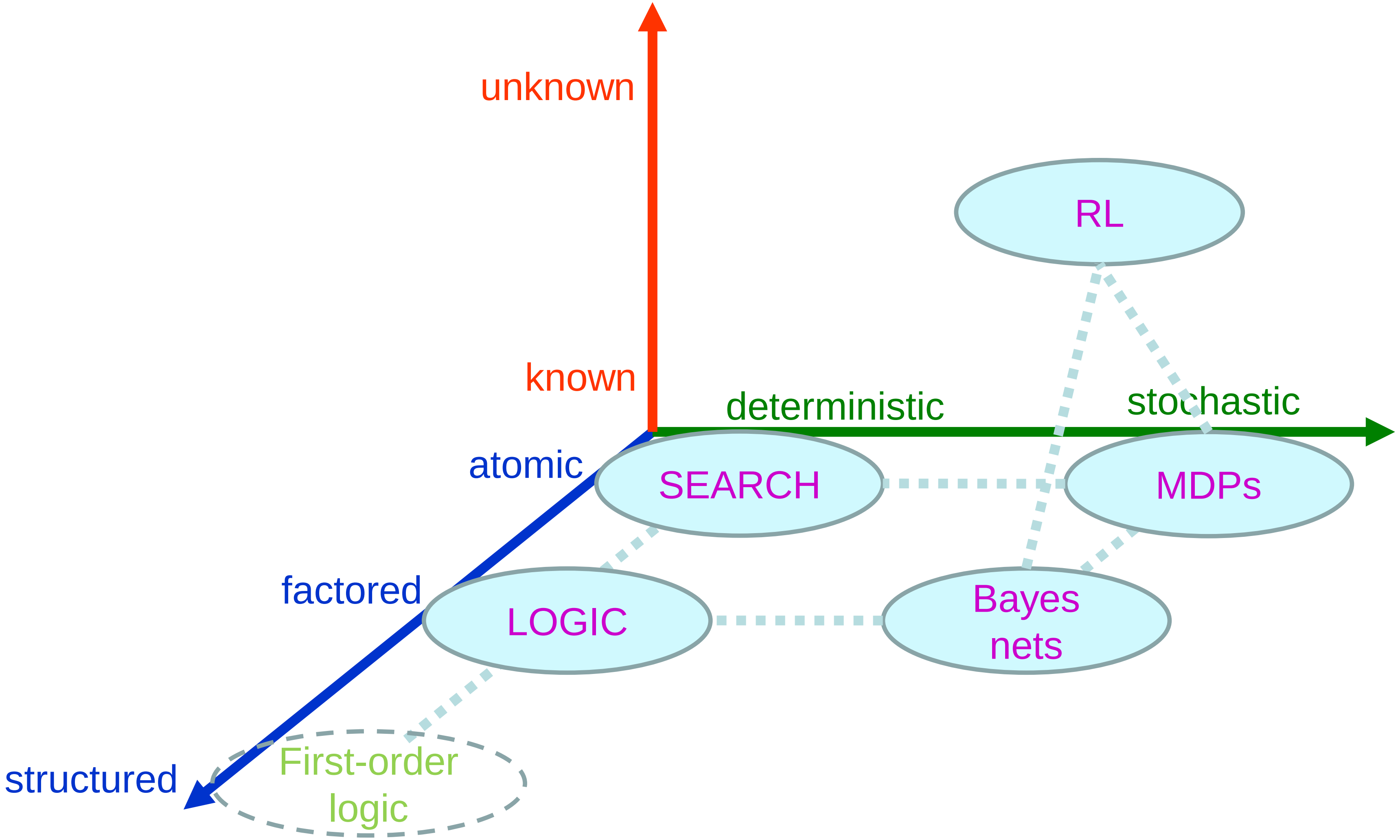
Artificial Intelligence

First Order Predicate Logic



Outline

- Limitation of Propositional Logic
- Introduction to First Order Logic (FOL)
- Knowledge Representation in FOL
  - Syntax
  - Semantics
- Logical inference in FOL
  - Inference Rule
  - Resolution
  - Forward and Backward Chaining

Pros and Cons of Propositional Logic

- Pros:
  - Propositional logic is **declarative**: pieces of syntax correspond to facts
  - Propositional logic allows **partial**/**disjunctive**/**negated** information (unlike most data structures and databases)
  - Propositional logic is **compositional**: meaning of $B \wedge P$ is derived from meaning of B and of P
  - Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- Cons:
  - Propositional logic has very **limited expressive power** (unlike natural language)

Limitation of Propositional Logic

- Propositional logic has limited expressive power.
- We cannot represent relations like All, Some, or none with propositional logic. Example:
  - Statements about similar objects, relations
  - Statements referring to groups of objects
    - **All the animals are intelligent.**
    - **Some apples are sweet.**
- We cannot describe statements in terms of their properties or logical relationships.

Limitation of Propositional Logic

- Statements referring to groups of objects
- Example:
  - Assume we want to express: "Every student likes vacation"
  - Require to include statements about every student

    Rohit likes vacation ∧

    Virat likes vacation ∧

    Shubham likes vacation ∧ …..
- Problem: KB grows large
- Solution: ?
- Possible solution:
  - Allow quantification in statements
  - Universal (∀) and Existential (∃)

- Statements about objects and relations
- Example:
  - Seniority of people domain
    - Virat is older than Rohit
    - Rohit is older than Shubham
  - To derive: Virat is older than Shubham

    Virat is older than Rohit ∧ Rohit is older than Shubham ⇒ Virat is older than Shubham
  - Assume we add another fact:

    Mahi is older than Rohit
  - To derive:

    Mahi is older than Rohit ∧ Rohit is older than Shubham ⇒ Mahi is older than Shubham
- Possible solution: introduce variables

A is older than **B** ∧ **B** is older than **C** ⇒ **A** is older than **C**

$\forall x$ student(x) ⇒ like_vacation(x)

First-Order Logic

- Another way of knowledge representation in artificial intelligence.
- FOL is sufficiently expressive to represent the natural language statements
- **Propositional logic**: world contains facts
- **First-order logic**: the world contains objects, relations, and functions
  - **Objects**: people, houses, numbers, theories, location, colors, baseball games, wars, centuries . . .
  - **Relations**: 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, end of . . .
- Introducing variables that refer to an arbitrary objects
- Introducing quantifiers allowing us to make statements over groups objects

Logic

- Logic is defined by:
  - A set of sentences
    - A sentence is constructed from a set of primitives according to syntax rules.
  - A set of interpretations
    - An interpretation gives a semantic to primitives. It associates primitives with objects, values in the real world.
  - The valuation (meaning) function *V*
    - Assigns a truth value to a given sentence under some interpretation
    - V : sentence × interpretation → {*True* , *False* }

First Order Logic: Syntax

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
& \mid & \neg\ \mathit{Sentence} \\
& \mid & \mathit{Sentence} \land \mathit{Sentence} \\
& \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
& \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
& \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
& \mid & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
\\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
& \mid & \mathit{Constant} \\
& \mid & \mathit{Variable} \\
\\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

- Term: syntactic entity for representing objects
  - **Constant symbols:** represent specific objects

    Virat, India, Car
  - **Variables:** represent objects of a certain type

    x, y, z
  - **Functions:** applied to one or more terms

    father-of (Pratik), father-of(father-of(Pratik))

First Order Logic: Syntax

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate} \mid \mathit{Predicate}(\mathit{Term},\ldots) \mid \mathit{Term} = \mathit{Term} \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
& \mid & \neg\, \mathit{Sentence} \\
& \mid & \mathit{Sentence} \land \mathit{Sentence} \\
& \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
& \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
& \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
& \mid & \mathit{Quantifier}\ \mathit{Variable},\ldots\ \mathit{Sentence} \\
\\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term},\ldots) \\
& \mid & \mathit{Constant} \\
& \mid & \mathit{Variable} \\
\\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

- Sentences in FOL:
  - Atomic sentences:
    - A predicate symbol applied to 0 or more terms
  - Examples:
    - *Red*(*Rose*)
    - *Sister*(*Arati, Jyoti*)
    - *Manager*(*father-of*(*Ritesh*)
  - t1 = t2 equivalence of terms
  - Example:
    - *Chha. Shivaji Maharaj = father-of*(Chha. *Sambhaji Maharaj*)

First Order Logic: Syntax

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \land \textit{Sentence} \\
& \mid & \textit{Sentence} \lor \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Rightarrow\ \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Leftrightarrow\ \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

- Sentences in FOL:
  - Complex sentences:
  - Assume P, Q are sentences in FOL. Then:
    $(P \land Q)$, $(P \lor Q)$, $(P \Rightarrow Q)$, $(P \Leftrightarrow Q)$, $\neg Q$
    and $\forall x\, P$ $\exists y\, Q$ are sentences
- Symbols $\exists$, $\forall$
  - stand for the **existential** and the **universal** quantifier

First Order Logic: Syntax

$$
\begin{array}{rcl}
Sentence & \rightarrow & AtomicSentence \mid ComplexSentence \\
AtomicSentence & \rightarrow & Predicate \mid Predicate(Term, \ldots) \mid Term = Term \\
ComplexSentence & \rightarrow & (\ Sentence\ ) \mid [\ Sentence\ ] \\
& \mid & \neg\ Sentence \\
& \mid & Sentence \land Sentence \\
& \mid & Sentence \lor Sentence \\
& \mid & Sentence \ \Rightarrow\ Sentence \\
& \mid & Sentence \ \Leftrightarrow\ Sentence \\
& \mid & Quantifier\ Variable, \ldots\ Sentence \\
\\
Term & \rightarrow & Function(Term, \ldots) \\
& \mid & Constant \\
& \mid & Variable \\
\\
Quantifier & \rightarrow & \forall \mid \exists \\
Constant & \rightarrow & A \mid X_1 \mid John \mid \cdots \\
Variable & \rightarrow & a \mid x \mid s \mid \cdots \\
Predicate & \rightarrow & True \mid False \mid After \mid Loves \mid Raining \mid \cdots \\
Function & \rightarrow & Mother \mid LeftLeg \mid \cdots \\
\textsc{Operator Precedence} & : & \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

- All birds fly.
  - The predicate is "*fly*(*bird*).“
  - $\forall x\ bird(x) \Rightarrow fly(x)$
- Every man respects his parent.
  - The predicate is "*respect*(*x*, *y*)," where *x*=man, and *y*= parent.
  - $\forall x\ man(x) \Rightarrow respects(x, parent)$
- Some boys play cricket.
  - In this sentence, the predicate is "*play*(*x*, *y*)," where *x*= boys, and *y*= game.
  - $\exists x\ boys(x) \Rightarrow play(x, cricket)$

Dealing with Quantifiers

- $\forall$ can be thought of as "conjunction" over all objects in domain:
  - e.g., $\forall x\, bird(x)$
    can be interpreted as $bird(tweety) \wedge bird(sam) \wedge bird(fred) \wedge \ldots$
- $\exists$ can be thought of as "disjunction" over all objects in domain:
  - e.g., $\exists x\, bird(x)$
    can be interpreted as $bird(tweety) \vee bird(sam) \vee bird(fred) \vee \ldots$
- Quantifier Duality
  - Each can be expressed using the other
  - This is an application of DeMorgan's laws
  - examples:

$$\forall x\ loves(x, tweety) \quad \text{is equivalent to} \quad \neg\exists x \neg loves(x, tweety)$$

$$\exists x\ likes(x, broccoli) \quad \text{is equivalent to} \quad \neg\forall x \neg likes(x, broccoli)$$

Dealing with Quantifiers

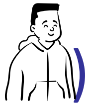
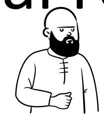
- Usually use $\Rightarrow$ with $\forall$:
  - e.g., $\forall x\; human(x) \Rightarrow mortal(x)$ says, all humans are mortal

    but, $\forall x\; human(x) \wedge mortal(x)$ say, everything is human and mortal
- Usually use $\wedge$ with $\exists$:
  - e.g., $\exists x\; bird(x) \wedge \neg flies(x)$ says, there is a bird that does not fly

    but, $\exists x\; bird(x) \Rightarrow \neg flies(x)$ is also true for anything that is not a bird
- $\forall x \exists y$ is not the same as $\exists y \forall x$:
  - e.g., $\exists x \forall y\; loves(x, y)$ says, there is someone who loves everyone

    but, $\forall y \exists x\; loves(x, y)$ says, everyone is loved by at least one person

Semantic: Interpretation

- An interpretation ***I*** is defined by a mapping to the domain of discourse ***D*** or relations on ***D***
  - Domain of discourse: a set of objects in the world we represent and refer to;
- An interpretation ***I*** maps:
  - Constant symbols to objects in D
    - *I*(*Sachin*) = 
  - Predicate symbols to relations, properties on D
    - *I*(*brother*) = { (, ); (, ); ……. }
  - Function symbols to functional relations on D
    - *I*(*father-of*) = {( → ); {( → ); ……….}

Semantics of Sentences

- Meaning (evaluation) function:
  - V : sentence × interpretation → {*True* , *False*}
- A predicate: predicate(*term*-1, *term*-2, *term*-3, *term-n*) is true for the interpretation *I* , iff the objects referred to by *term*-1, *term*-2, *term*-3, *term-n* are in the relation referred to by predicate

- *I*(*Rohit*) =  *I*(*Kunal*) = 

- *I*(*brother*) = { ( , ); ( , ); ……. }

- *brother*(*Rohit*, *Kunal*) = ( , ) in *I*(*brother*)

- *V*(*brother*(*Rohit*, *Kunal*), *I*) = *True*

Semantics of Sentences with quantifiers

- **Universal quantification**
- $V(\forall x\ \emptyset, I) = True$

  substitution of x with d

  - Iff for all $d \in D$, $V(\emptyset, I[x/d]) = True$

- **Existential quantification**
- $V(\exists x\ \emptyset, I) = True$
  - Iff for all $d \in D$, s.t. $V(\emptyset, I[x/d]) = True$

Semantics of Sentences with quantifiers

- **Universal quantification**
  - All COEP students are smart
- Assume the universe of discourse of x are *COEP students*
  - $\forall x\ at(x, COEP) \Rightarrow smart(x)$
- Assume the universe of discourse of x are *students*
  - $\forall x\ smart(x)$
- Assume the universe of discourse of x are *people*
  - $\forall x\ student(x) \wedge at(x, COEP) \Rightarrow smart(x)$

- **Existential quantification**
  - Someone at COEP is smart
- Assume the universe of discourse of x are *COEP* affiliates
  - $\exists x\ smart(x)$
- Assume the universe of discourse of x are *people*
  - $\exists x\ at(x, COEP) \wedge student(x)$

Semantics of Sentences with quantifiers

Assume two predicates S(x) and P(x)

- **Universal quantification**
- Typically, the ∀ connects with implication
- All S(x) is P(x)
  - $\forall x \,(S(x) \Rightarrow P(x))$
- No S(x) is P(x)
  - $\forall x \,(S(x) \Rightarrow \neg P(x))$

- **Existential quantification**
- Typically, the ∃ connects with a conjunction
- Some S(x) is P(x)
  - $\exists x \,(S(x) \land P(x))$
- Some S(x) is not P(x)
  - $\exists x \,(S(x) \land \neg P(x))$

Connection Between Quantifiers

- Everyone likes ice cream
  - $\forall x \; likes(x, IceCream)$
- Is it possible to convey the same meaning using an existential quantifier ?
  - There is no one who does not like ice cream
  - $\neg\exists x \; \neg likes(x, IceCream)$
- A universal quantifier in the sentence can be expressed using an existential quantifier !!!

Connection Between Quantifiers

- Someone likes ice cream
  - $\exists x\ likes(x, IceCream)$

- Is it possible to convey the same meaning using an Universal quantifier ?
  - Not everyone does not like ice cream
  - $\neg\forall x\ \neg likes(x, IceCream)$

- A existential quantifier in the sentence can be expressed using an universal quantifier !!!

Nested quantifiers

- More than one quantifier may be necessary to capture the meaning of a statement in the predicate logic.
- Order of quantifiers of the same type does not matter
  - For all *x* and *y*, if *x* is a parent of *y* then *y* is a child of *x*
    - $\forall x,y\ parent(x, y) \Rightarrow child(y, x)$
    - $\forall y,x\ parent\ (x, y) \Rightarrow child(y, x)$
- Order of different quantifiers changes the meaning
  - $\forall x \exists y\ loves(x, y)$: Everybody loves somebody
  - $\exists y \forall x\ loves(x, y)$: There is someone who is loved by everyone

Examples

- Suppose:
  - Variables x, y denote people
  - Loves(x, y) denotes "x loves y"

- Translate:
  - Everybody loves Virat.
  - Everybody loves somebody.
  - There is somebody whom everybody loves.
  - There is somebody who Virat doesn't love.
  - There is somebody whom no one loves.

  - $\forall x$ Loves(x, *Virat*)
  - $\forall x \exists y$ Loves(x, y)
  - $\exists y \forall x$ Loves(x, y)
  - $\exists y \neg$Loves(*Virat*, y)
  - $\exists y \forall x \neg$Loves(x, y)

Some examples of FOL sentences

- Every gardener likes the sun.

  $\forall x\ gardener(x) \Rightarrow likes\ (x, Sun)$

- You can fool some of the people all of the time

  $\exists x\ (person(x) \wedge \forall\ t\ (time(t) \Rightarrow can\text{-}fool(x, t)))$

- You can fool all of the people some of the time.

  $\forall x\ (person(x) \Rightarrow \exists\ t\ (time(t) \wedge can\text{-}fool(x, t)))$

- No purple mushroom is poisonous.

  $\neg\exists x\ purple(x) \wedge mushroom(x) \wedge poisonous(x)$

  or, equivalently,

  $\forall x\ (mushroom(x) \wedge purple(x)) \Rightarrow \neg poisonous(x)$

Transformation to FOPC

$goodgrade(mary, cs101) \wedge goodgrade(mary, cs102)$

Mary got good grades in courses CS101 and CS102

$pass(john, cs102)$

John passed CS102

$\forall x, y[student(x) \wedge course(y) \wedge goodgrade(x, y) \Rightarrow pass(x, y)]$

Student who gets good grades in a course passes that course

$\forall x[student(x) \wedge \exists y[course(y) \wedge pass(x, y)] \Rightarrow happy(x)]$

Students who pass a course are happy

$\forall x[student(x) \wedge \neg happy(x) \Rightarrow \exists y[course(y) \wedge \neg pass(x, y)]]$

A student who is not happy hasn’t passed all his/her courses

$$\exists x[student(x) \wedge \forall y[course(y) \Rightarrow \neg pass(x, y)]$$
$$\wedge \forall z[(student(z) \wedge \neg(x = z)) \Rightarrow \exists y[course(y) \wedge pass(z, y)]]$$

Only one student failed all the courses

Representing knowledge in FOL

Example: Kinship domain

- Objects: people

  Jyoti, Arti, Dipak, …

- Properties: gender

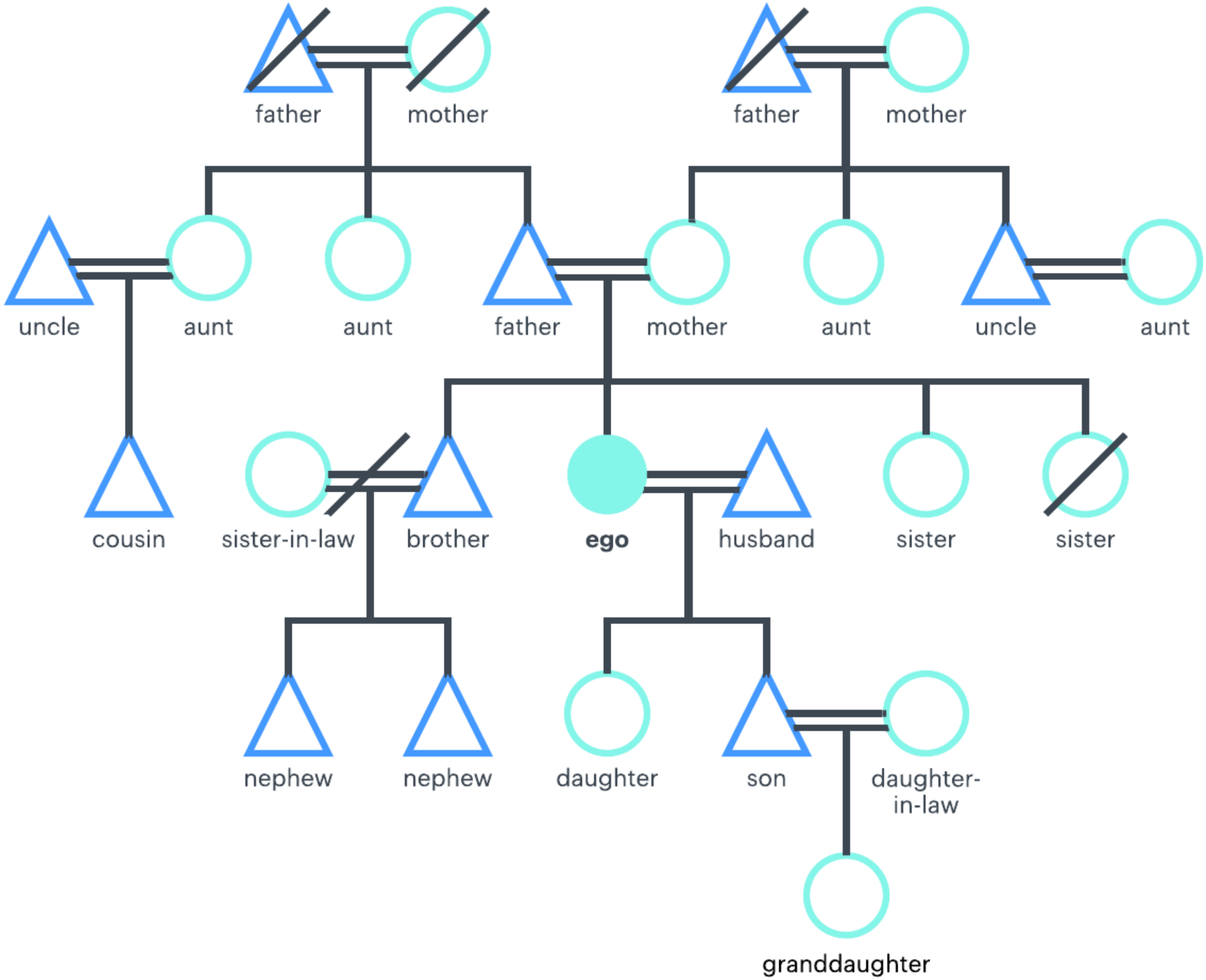
  Male(x), Female(x)

- Relations: parenthood, brotherhood, marriage

  Parent (x, y), Brother (x, y), Spouse (x, y)

- Functions: mother-of (one for each person x)

  MotherOf (x)

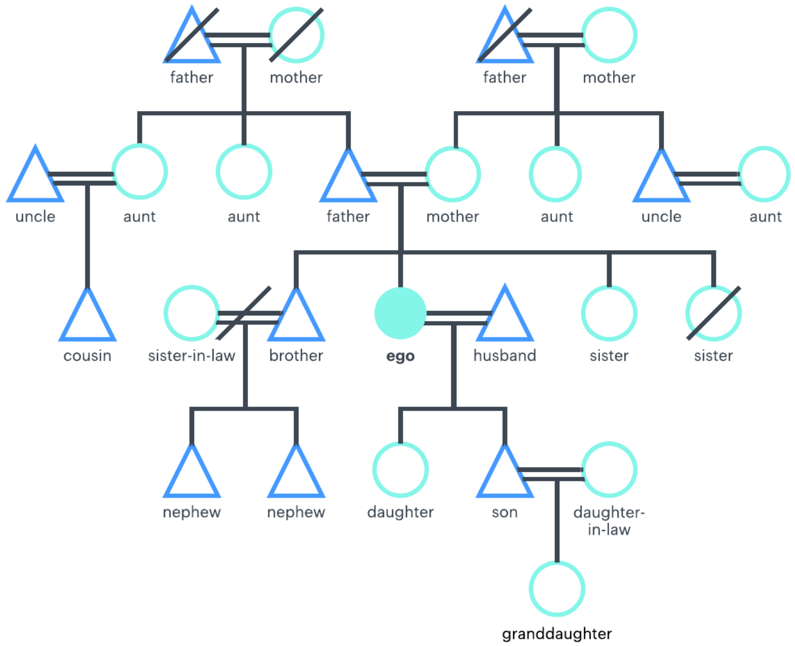


Representing knowledge in FOL

- Family
  - Spouse
    - Husband
    - Wife
  - Parent
    - Father
    - Mother
    - Step-father
    - Step-mother
    - Legal guardian
  - Child
    - Son
    - Daughter
    - Step-son
    - Step-daughter
  - Sibling
    - Brother
    - Sister

- Extended family
  - Grandparent
  - Grandfather
  - Grandmother
  - Grandson
  - Granddaughter
  - Uncle
  - Aunt
  - Cousin
  - Nephew
  - Niece
- Family-in-law
  - Father-in-law
  - Mother-in-law
  - Brother-in-law
  - Sister-in-law

- Relation change
  - Marriage
  - Adoption
  - Relation end
    - Breakup
    - Divorced
    - Disownment
    - Emancipation
    - Widowhood



Representing knowledge in FOL

Relations between predicates and functions: write down what we know about them; how relate to each other.

- Male and female are disjoint categories

  $\forall x \; Male(x) \Leftrightarrow \neg Female(x)$

- Parent and child relations are inverse

  $\forall x, y \; Parent(x, y) \Leftrightarrow Child(y, x)$

- A grandparent is a parent of parent

  $\forall g, c \; Grandparent(g, c) \Leftrightarrow \exists p \; Parent(g, p) \wedge Parent(p, c)$

- A sibling is another child of one's parents

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

- And so on …

Knowledge Representation

INFERENCE IN FIRST ORDER LOGIC

Logical inference in FOL

Logical inference problem:

- Given a knowledge base **KB** (a set of sentences) and **α** (a sentence) , does the **KB** semantically entail **α**?

  KB $\models$ α ?

- In other words: In all interpretations in which sentences in the **KB** are true, is also **α** true ?

- Logical inference problem in the first-order logic is undecidable !!!.
- No procedure that can decide the entailment for all possible input sentences in a finite number of steps.

Logical inference in FOL

- Logical Inference in PL

  KB $\models$ α ?

- Three approaches:
  - Truth-table approach
  - Inference rules
  - Resolution-refutation

- Truth-table approach a viable for the FOL?
- **NO!**
- It would require us to enumerate and list all possible interpretations I
- *I* = (assignments of *symbols* to *objects*, *predicates* to *relations* and *functions* to *relational* mappings)
- Simply there are too many interpretations

- Inference rule approach a viable for the FOL?
- **Yes.**
- The inference rules represent *sound inference* patterns one can apply to sentences in the KB
- What is derived follows from the KB
- Caveat: Need to add rules for handling quantifiers
- Must involve variable substitutions

Theorem Proving: Inference Rule

- (MP) Modes Ponens $$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$
- (AI) And-Introduction $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$
- (OI) Or-Introduction $$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n}$$
- (AE) And-Elimination $$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$
- (NE) Negation-Elimination $$\frac{\neg\neg\alpha}{\alpha}$$
- (UR) Unit Resolution $$\frac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$$
- (R) General Resolution $$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Additional inference rules are needed
  - For sentences with *quantifiers* and *variables*
  - Must involve variable substitutions

Variable Substitutions

- Variables in the sentences can be substituted with terms.

  (terms = constants, variables, functions)

- Substitution: Is a mapping from *variables* to *terms*

  $\{x_1/t_1, x_2/t_2, \ldots..\}$

- Application of the substitution to sentences

  SUBST({*x/Ram, y/Sham*}, Likes(*x, y*)) = Likes(*Ram, Sham*)

  SUBST({ *x/Ram , y/fatherof*(*Ram*)}, Likes(*x, y*)) = Likes(*Ram, fatherof*(*Ram*))

Inference Rules for Quantifiers

Universal Instantiation/elimination

$$\frac{\forall x\ \emptyset(x)}{\emptyset(a)}$$

a is constant

- Substitutes a variable with a constant symbol
- Example:

  $\forall x$ *Likes*(*x*, *IceCream*) → *Likes*(*Virat*, *IceCream*)

Existential Instantiation/elimination

$$\frac{\exists x\ \emptyset(x)}{\emptyset(a)}$$

- Substitutes a variable with a constant symbol
- that symbol does not appear elsewhere in the KB
- Example:

  Special constant called a ***Skolem*** constant

  $\exists x$ *Kill*(*x*, *Victim*) → *Kill*(*Murderer*, *Victim*)

  $\exists y \forall x$(*likes*(*x*, *y*)) → $\forall x$ *likes*(*x*, *a*)

  Special function called a ***Skolem*** function

  $\forall x \exists y$ (*likes*(*x*, *y*)) → $\forall x$ *likes*(*x*, *f*(*x*))

Example of Derivation

- Let KB = $\{ parent(john, mary),\ parent(john, joe),\ \forall x \forall y\, [\exists z\, (parent(z,x) \wedge parent(z,y) \Rightarrow sibling(x,y))\,] \}$
  $KB \models sibling(mary, joe)$

1. $\forall x \forall y\, [\exists z\, (parent(z,x) \wedge parent(z,y) \Rightarrow sibling(x,y))\,]$ (from KB)
2. $\forall y\, [\exists z\, (parent(z,mary) \wedge parent(z,y) \Rightarrow sibling(mary,y))\,]$ (1, UI)
3. $\exists z\, (parent(z,mary) \wedge parent(z, joe) \Rightarrow sibling(mary, joe))$ (2, UI)
4. $parent(john,mary) \wedge parent(john, joe) \Rightarrow sibling(mary, joe)$ (3, EI)
5. $parent(john,mary)$ (from KB)
6. $parent(john, joe)$ (from KB)
7. $parent(john,mary) \wedge parent(john, joe)$ (5, 6, AI)
8. $sibling(mary, joe)$ (4, 7, MP)

- This derivation shows that $KB \models sibling(mary, joe)$

Reduction to Propositional Inference

- Suppose the KB contains just the following:

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

  King(John)

  Greedy(John)

  Brother(Richard, John)

- Instantiating the universal sentence in all possible ways, we have:

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

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

  King(John)

  Greedy(John)

  Brother(Richard, John)

- The new KB is propositionalized: proposition symbols are

  King(John), Greedy(John), Evil(John), King(Richard), etc.

Problems with Propositionalization

- Propositionalization seems to generate lots of irrelevant sentences
- E.g., 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 facts such as *Greedy(Richard)* that are irrelevant

Unification (alignment)

- Problem in inference: Universal elimination gives many opportunities for substituting variables with ground terms
- $\dfrac{\forall x\ \emptyset(x)}{\emptyset(a)}$ a is constant
- Solution: Try substitutions that help us to make progress
  - Use substitutions of “similar” sentences in KB
- Example:
  - ∀x King(x) ∧ Greedy(x) ⇒ Evil(x)
  - King(John)
  - ∀y Greedy(y)
  - If we use a substitution σ = {x/John, y/John}
  - we can use modus ponens to infer Evil(John) in one step

Unification

- Takes two similar sentences and computes the substitution that makes them look the same, if it exists
  - Use substitutions of “similar” sentences in KB

$UNIFY\ (p, q) = \sigma$ such that $SUBST(\sigma, p) = SUBST\ (\sigma, q)$

- Examples:

UNIFY (*Knows*(*John*, *x*), *Knows*(*John*, *Jane*)) = {*x*/*Jane*}

UNIFY(*Knows*(*John*, *x*),*Knows*(*y*, *Ann*)) = {*x*/*Ann*, *y*/*John*}

UNIFY(*Knows*(*John*, *x*),*Knows*(*y*, *Motherof*(*y*))) = {*x*/*Motherof*(*John*), *y*/*John*}

UNIFY(*Knows*(*John*, *x*), *Knows*(*x*, *Elizabeth*)) = *fail*

Unification

- To unify Knows(John,x) and Knows(y,z),

  σ = {y/John, x/z } or σ = {y/John, x/John, z/John}

- The **first** unifier is **more general** than the **second**.
- There is a single most general unifier (MGU) that is unique up to renaming of variables.

  MGU = { y/John, x/z }

Generalized Inference Rules

- Use substitutions that let us make inferences
- Example: **Modus Ponens**
- If there exists a substitution σ such that

$SUBST(\sigma, A_i) = SUBST(\sigma, A_i)$ **For all i = 1, 2, 3, …, n**

$$\frac{(A_1 \land A_2 \land \ldots . A_n \Rightarrow B, \quad A'_1, \quad A'_2, \quad \ldots A'_n)}{SUBST(\sigma, \quad B)}$$

- Substitution that satisfies the generalized inference rule can be build via unification process

Resolution inference rule

- Recall: Resolution inference rule is sound and complete (refutation-complete) for the propositional logic and CNF

Resolution

$$\frac{\begin{array}{c} p \lor q \\ \neg p \lor r \end{array}}{\text{Therefore, } q \lor r}$$

- Generalized resolution rule is sound and refutation complete for the first-order logic and CNF w/o equalities (if unsatisfiable the resolution will find the contradiction)

$$\sigma = UNIFY\ (\phi_i, \neg\psi_j) \neq fail$$

$$\frac{\phi_1 \lor \phi_2 \ldots \lor \phi_k, \quad \psi_1 \lor \psi_2 \lor \ldots \psi_n}{SUBST(\sigma, \phi_1 \lor \ldots \lor \phi_{i-1} \lor \phi_{i+1} \ldots \lor \phi_k \lor \psi_1 \lor \ldots \lor \psi_{j-1} \lor \psi_{j+1} \ldots \psi_n)}$$

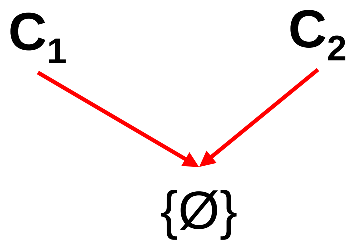
- Example

$$\frac{P(x) \lor Q(x), \quad \neg Q(John) \lor S(y)}{P(John) \lor S(y)}$$

Resolution Rule

- Proof by refutation:
  - To prove that $KB \models \alpha$, show that $\boldsymbol{KB} \wedge \neg\alpha$ is ***unsatisfiable***
  - ***KB*** and $\neg\alpha$, must be in **CNF** (conjunction of clauses)
  - Resolution is **refutation-complete**

- Refutation procedure
  - Each step in the refutation procedure involves applying resolution to two clauses, in order to get a new clause
  - Inference continues until the empty clause {Ø} is derived (a contradiction)



Recall the Refutation Procedure - Example

Given *KB* =
$$\left\{\begin{array}{ll} 1. & \neg A \vee \neg B \vee C \\ 2. & \neg D \vee B \\ 3. & \neg E \vee A \\ 4. & E \\ 5. & D \end{array}\right\}$$

Prove *KB* ⊨ *C*

$\neg C$ $\quad$ $\neg A \vee \neg B \vee C$ **(clause 1)**

$\neg A \vee \neg B$ $\quad$ $\neg E \vee A$ **(clause 3)**

$\neg E \vee \neg B$ $\quad$ E **(clause 4)**

$\neg B$ $\quad$ $\neg D \vee B$ **(clause 2)**

$\neg D$ $\quad$ D **(clause 5)**

$\boxtimes$

Resolution Rule of Inference

- Basic Propositional Version:

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \textbf{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

- Full First-Order Version:

$$\frac{(p_1 \vee \ldots \vee p_j \vee \ldots \vee p_m),\, (q_1 \vee \ldots \vee q_k \vee \ldots \vee q_n)}{(p_1 \vee \ldots \vee p_{j-1} \vee p_{j+1} \vee \ldots \vee p_m \vee q_1 \vee \ldots \vee q_{k-1} \vee q_{k+1} \vee \ldots \vee q_n)\sigma}$$

provided that p_j and $\neg q_k$ are *unifiable* via a *substitution* σ

- Example:

$$\frac{\neg rich(x) \vee unhappy(x) \qquad rich(bob)}{unhappy(bob)}$$

with substitution $\sigma = \{x/bob\}$

Resolution Example

KB = {¬P(w) ∨ Q(w)

¬Q(y) ∨ S (y)

P(x) ∨ R x)

¬R(z) ∨ S (z) }

α = S(A)

Resolution Example

KB **¬ α**

¬P(w) V Q(w) ¬Q(y) V S (y) P(x) V R x) ¬R(z) V S (z), ¬S(A)

{y / w}

¬P(w) V S (w)

{x / w}

S(w) V R(w)

{z / w}

S (w)

{w / A}

{ }

Contradiction

KB ⊨ α

Conversion to CNF

1. Eliminate implications, equivalences

$$(p \Rightarrow q) \rightarrow (\neg p \vee q)$$

2. Move negations inside (DeMorgan's Laws, double negation)

$$\neg(p \wedge q) \rightarrow \neg p \vee \neg q$$
$$\neg(p \vee q) \rightarrow \neg p \wedge \neg q$$
$$\neg \forall x \; p \rightarrow \exists x \; \neg p$$
$$\neg \exists x \; p \rightarrow \forall x \; \neg p$$
$$\neg\neg \; p \rightarrow p$$

3. Standardise variables (Rename duplicate variables)

$$(\forall x \; P(x)) \vee (\exists x \; Q(x)) \rightarrow (\forall x \; P(x)) \vee (\exists y \; Q(y))$$

4. Move all quantifiers left (no invalid capture possible)

$$(\forall x \; P(x)) \vee (\exists y \; Q(y)) \rightarrow \forall x \; \exists y \; P(x) \vee Q(y)$$

Conversion to CNF

5. Skolemization (removal of existential quantifiers through elimination)
   - If no universal quantifier occurs before the existential quantifier, replace the variable with a new constant symbol

$$\exists y\ P(A) \vee Q(y) \rightarrow P(A) \vee Q(B)$$

   - If a universal quantifier precede the existential quantifier replace the variable with a function of the "universal" variable

$$\forall x\ \exists y\ P(x) \vee Q(y) \rightarrow \forall x\ \ P(x) \vee Q(F(x))$$

F(*x*) - a Skolem function

6. Drop universal quantifiers (all variables are universally quantified)

$$\forall x\ \ P(x) \vee Q(F(x)) \rightarrow P(x) \vee Q(F(x))$$

7. Convert to CNF using the distributive laws

$$p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)$$

Conversion to CNF - Example

Convert: $\forall x\,[(\forall y\; p(x,y)) \Rightarrow \neg(\forall y\,(q(x,y) \Rightarrow r(x,y)))]$

(1) $\forall x\,[\neg(\forall y\; p(x,y)) \vee \neg(\forall y\,(\neg q(x,y) \vee r(x,y)))]$ — Replace $P \Rightarrow Q$ with $\neg P \vee Q$

(2) $\forall x\,[(\exists y\; \neg p(x,y)) \vee (\exists y\,(q(x,y) \wedge \neg r(x,y)))]$ — Move inward the negation symbol

(3) $\forall x\,[(\exists y\; \neg p(x,y)) \vee (\exists z\,(q(x,z) \wedge \neg r(x,z)))]$ — Standardize variables apart

(4) $\forall x \exists y \exists z\,[\neg p(x,y) \vee (\,q(x,z) \wedge \neg r(x,z))]$ — Move quantifiers left in order

(5) $\forall x\,[\neg p(x,sk_1(x)) \vee (\,q(x,sk_2(x)) \wedge \neg r(x,sk_2(x)))]$ — Eliminate ∃ by Skolemization

(6) $\neg p(x,sk_1(x)) \vee (\,q(x,sk_2(x)) \wedge \neg r(x,sk_2(x)))$ — Drop universal quantifiers

(7) $[\neg p(x,sk_1(x)) \vee q(x,sk_2(x))] \wedge [\neg p(x,sk_1(x)) \vee \neg r(x,sk_2(x))]$ — Distribute ∧ over ∨

(8) $\{\, \neg p(x,sk_1(x)) \vee q(x,sk_2(x)),\quad \neg p(w,sk_1(w)) \vee \neg r(w,sk_2(w))\,\}$ — Split conjunctions (into a set of clauses) and rename variables

Refutation Procedure - Example

$$
KB = \left\{
\begin{array}{ll}
1. & father(john, mary) \\
2. & mother(sue, john) \\
3. & father(bob, john) \\
4. & \forall x \forall y[(father(x,y) \vee mother(x,y)) \Rightarrow parent(x,y)] \\
5. & \forall x \forall y[\exists z(parent(x,z) \wedge parent(z,y)) \Rightarrow grand(x,y)]
\end{array}
\right\}
$$

$$KB \models \exists x \; parent(x, john)$$

$$KB \models grand(sue, mary)$$

Refutation Procedure - Example

$$
\textbf{KB} = \left\{
\begin{array}{ll}
1. & father(john, mary) \\
2. & mother(sue, john) \\
3. & father(bob, john) \\
4. & \forall x \forall y [(father(x,y) \vee mother(x,y)) \Rightarrow parent(x,y)] \\
5. & \forall x \forall y [\exists z (parent(x,z) \wedge parent(z,y)) \Rightarrow grand(x,y)]
\end{array}
\right\}
$$

Converting 4 to CNF:

$$4. \quad (\neg father(x,y) \vee parent(x,y)) \wedge (\neg mother(x,y) \vee parent(x,y))$$

Converting 5 to CNF:

$$5. \quad \forall x \forall y [\neg \exists z (parent(x,z) \wedge parent(z,y)) \vee grand(x,y)]$$

$$\equiv \forall x \forall y \forall z [\neg (parent(x,z) \wedge parent(z,y)) \vee grand(x,y)]$$

$$\equiv \neg parent(x,z) \vee \neg parent(z,y) \vee grand(x,y)$$

Refutation Procedure - Example (cont.)

$$\textbf{\textit{KB}} = \left\{ \begin{array}{ll} 1. & father(john, mary) \\ 2. & mother(sue, john) \\ 3. & father(bob, john) \\ 4. & \neg father(x,y) \vee parent(x,y) \\ 5. & \neg mother(x,y) \vee parent(x,y) \\ 6. & \neg parent(x,z) \vee \neg parent(z,y) \vee grand(x,y) \end{array} \right\}$$

Here is the final KB in clausal form:

Next we want to prove the following using resolution refutation:

$\exists x \; parent(x, john)$ (there is someone who is john's parent)

Refutation Procedure - Example (cont.)

To prove, we must first negate the goal and transform into clausal form:

$$\neg\exists x\ parent(x, john) \longrightarrow \forall x\ \neg parent(x, john) \longrightarrow \neg parent(x, john)$$

The refutation (proof by contradiction):

$\neg parent(x, john)$ $\qquad$ $\neg father(x, y) \vee parent(x, y)$ **(clause 4)**

y* = *john

$\neg father(x, john)$ $\qquad$ $father(bob, john)$ **(clause 3)**

x* = *bob

$\boxtimes$

Note that the proof is *constructive*: we end up with an *answer* *x* = *bob*

Refutation Procedure - Example (cont.)

$$\textbf{\textit{KB}} = \left\{ \begin{array}{ll} 1. & father(john, mary) \\ 2. & mother(sue, john) \\ 3. & father(bob, john) \\ 4. & \neg father(x,y) \vee parent(x,y) \\ 5. & \neg mother(x,y) \vee parent(x,y) \\ 6. & \neg parent(x,z) \vee \neg parent(z,y) \vee grand(x,y) \end{array} \right\}$$

Here is the final KB in clausal form:

Next we want to prove the following using resolution refutation:

$grand(sue, mary)$ (sue is a grandparent of mary)

Refutation Procedure - Example (cont.)

Now, let's prove that *sue* is the grandparent of *mary*:

$\neg grand(sue, mary)$ $\quad$ $\neg parent(x,z) \vee \neg parent(z,y) \vee grand(x,y)$ **(clause 6)**

x = sue
y = mary

$\neg parent(sue,z) \vee \neg parent(z,mary)$ $\quad$ $\neg father(x_1,y_1) \vee parent(x_1,y_1)$ **(clause 4)**

$z = x_1$
y_1 = mary

$\neg parent(sue,x_1) \vee \neg father(x_1,mary)$ $\quad$ $father(john,mary)$ **(clause 1)**

x_1 = john

$\neg parent(sue, john)$ $\quad$ $\neg mother(x_2,y_2) \vee parent(x_2,y_2)$ **(clause 5)**

x_2 = sue
y_2 = john

$\neg mother(sue, john)$ $\quad$ $mother(sue, john)$ **(clause 2)**

$\boxtimes$

KB in Restricted Forms

- The sentences in the KB are restricted to some special forms
- **Horn form (Horn normal form)**

  (A ∨ ¬B) ∧ (¬ A ∨ ¬C ∨ D) Also be written as: (A ⇒ B) ∧ (A ∧ C) ⇒ D)

- A **disjunction of literals** with **at most one positive**, i.e. unnegated, literal.
- A clause with one literal, e.g. A, is also called a ***fact***
- A clause representing an implication (with a conjunction of positive literals in antecedent and one positive literal in consequent), is also called a ***rule***
- Two inference rules that are sound and complete with respect to propositional symbols for KBs in the Horn normal form:
  - Resolution (positive unit resolution)
  - Modus ponens

Resolution

$$\frac{p \vee q \qquad \neg p \vee r}{\text{Therefore, } q \vee r}$$

Modus Ponens

$$\frac{p \qquad p \rightarrow q}{\text{Therefore, } q}$$

Horn normal form in FOL

- First-order logic (FOL)
  - Adds variables, works with terms
- Generalized modus ponens rule:

$$\sigma = \text{a substitution s.t. } \forall i \; SUBST(\sigma, \phi_i') = SUBST(\sigma, \phi_i)$$

$$\frac{\phi_1', \phi_2' \ldots, \phi_n', \quad \phi_1 \wedge \phi_2 \wedge \ldots \phi_n \Rightarrow \tau}{SUBST\ (\sigma, \tau)}$$

- Generalized modus ponens:
  - Is sound and complete for definite clauses and no functions
  - In general it is semidecidable
  - Not all first-order logic sentences can be expressed in the HNF form

Entailment with Horn Clause

- Deciding entailment with Horn clauses can be done in time that is linear in the size of the knowledge base
  - Forward Chaining:
    - Idea: Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.
    - Typical usage: If we want to infer all sentences entailed by the existing KB.
  - Backward Chaining:
    - Idea: To prove the fact that appears in the conclusion of a rule prove the premises of the rule. Continue recursively
    - Typical usage: If we want to prove that the target (goal) sentence is α entailed by the existing KB.

Forward and Backward Chaining

- Work forward from **KB** to query **α:**
  - Fire any rule whose premises are satisfied in the KB
  - Add its conclusion to the KB, until query **α** is found
- Work backwards from the query **α** to **KB**:
  - Check if **α** is known already, or
  - Prove by BC all premises of some rule concluding **α**

- Assume the KB with the following rules and facts
  - **KB**:

    R1: Steamboat (x) ∧ Sailboat (y) ⇒ Faster (x, y)

    R2: Sailboat (y) ∧ RowBoat (z) ⇒ Faster (y, z)

    R3: Faster (x, y) ∧ Faster (y, z) ⇒ Faster (x, z)

    F1: Steamboat (*Titanic*)

    F2: Sailboat (*Mistral*)

    F3: RowBoat(*PondArrow*)

    α = Faster (*Titanic* , *PondArrow*)

Forward and Backward Chaining

- Assume the KB with the following rules and facts
  - **KB**: R1: $Steamboat(x) \land Sailboat(y) \Rightarrow Faster(x, y)$ $\alpha = Faster(Titanic, PondArrow)$

    R2: $Sailboat(y) \land RowBoat(z) \Rightarrow Faster(y, z)$

    R3: $Faster(x, y) \land Faster(y, z) \Rightarrow Faster(x, z)$

    F1: $Steamboat(Titanic)$

    F2: $Sailboat(Mistral)$

    F3: $RowBoat(PondArrow)$

    Rule R1 is satisfied: x/*Titanic*, y/*Mistral*

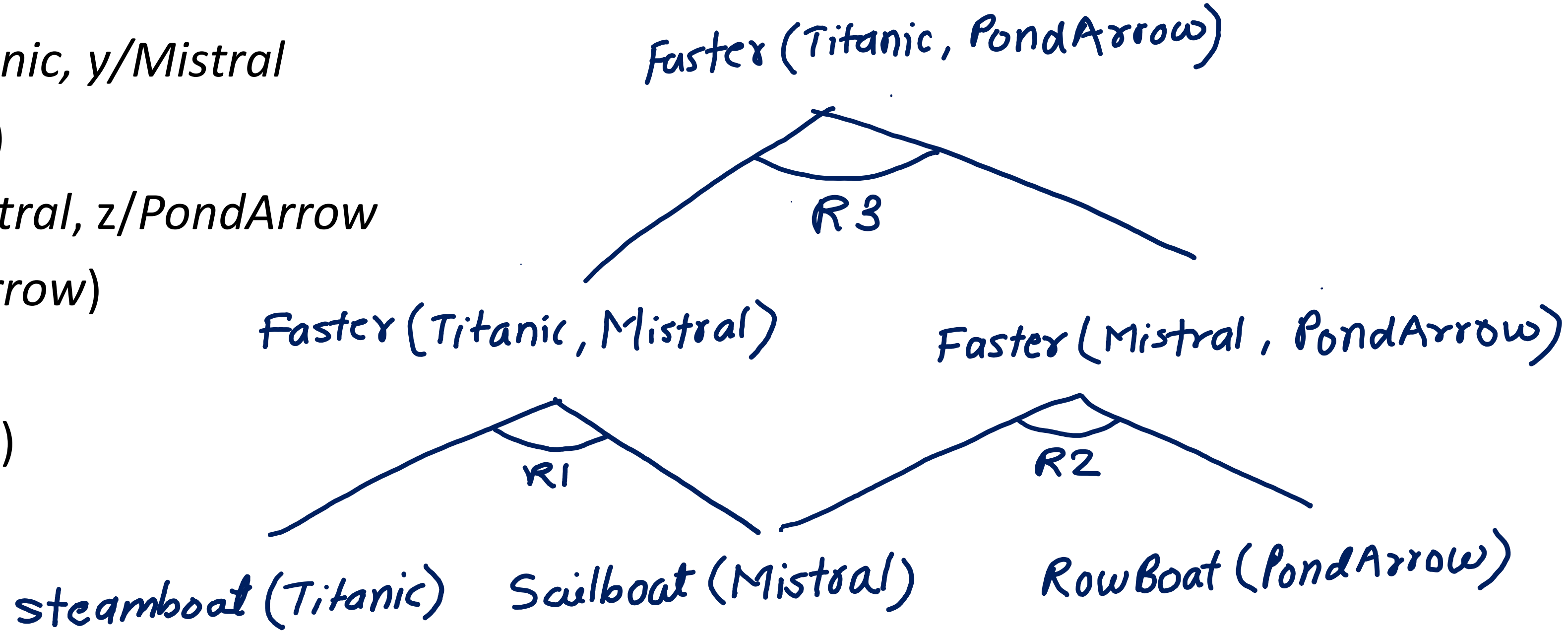
    F4: $Faster(Titanic, Mistral)$

    Rule R2 is satisfied: y/*Mistral*, z/*PondArrow*

    F5: $Faster(Mistral, PondArrow)$

    Rule R3 is satisfied:

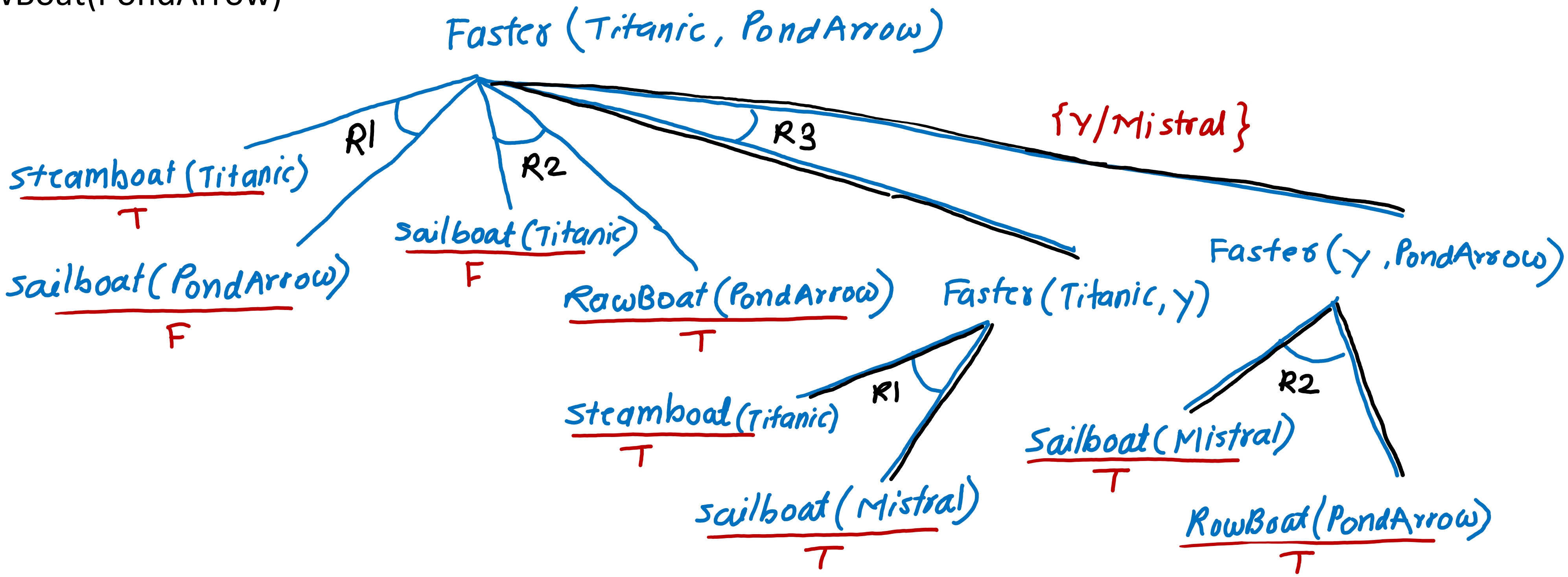
    $Faster(Titanic, PondArrow)$



Forward and Backward Chaining

- Assume the KB with the following rules and facts
  - **KB**:
    R1: Steamboat (x) ∧ Sailboat (y) ⇒ Faster (x, y)
    R2: Sailboat (y) ∧ RowBoat (z) ⇒ Faster (y, z)
    R3: Faster (x, y) ∧ Faster (y, z) ⇒ Faster (x, z)
    F1: Steamboat (Titanic)
    F2: Sailboat (Mistral)
    F3: RowBoat(PondArrow)

    α = Faster (Titanic , PondArrow)



Example

- Encode the following English statement into First-Order definite cluses.

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

- Prove the following by using the forward and backward chaining

  “Colonel West is a criminal”

Terms

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

- Constants:
  - America
  - Nano
  - West
- Function (propoerty):
  - American(x): x is an American
  - Weapon(x): x is a weapon
  - Hostile(x): x is a hostile nation
  - Criminal(x): x is a criminal
  - Missile(x): x is a missile
- Relation:
  - Owns(x, y): x owns y
  - Sells(x, y, z): x sells y to z
  - Enemy(x, y): x is an enemy of y

Example

- Sentences
  - It is a crime for an American to sell weapons to hostile nations
  - Nano ... has some missiles

  - All of its missiles were sold to it by Colonel West
  - Missiles are weapons
  - An enemy of America counts as "hostile“
  - West, who is American
  - The country Nano, an enemy of America

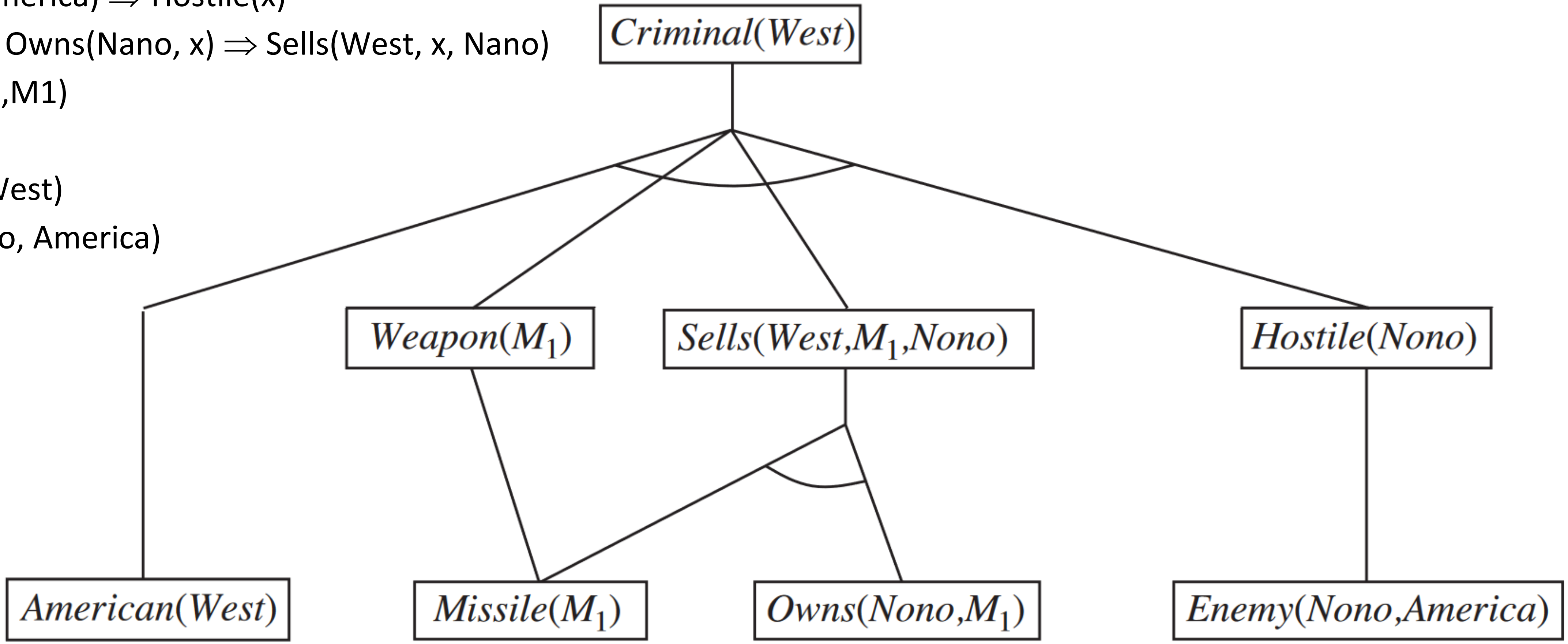
- First-Order Definite Clauses
  - $Sells(x, y, z) \wedge American(x) \wedge Weapon(y) \wedge Hostile(z) \Rightarrow Criminal(x)$
  - $\exists x\ Owns(Nano, x) \wedge Missile(x)$:

    $Owns(Nono,M1) \wedge Missile(M1)$
  - $Missile(x) \wedge Owns(Nano, x) \Rightarrow Sells(West, x, Nano)$

  - $Missile(x) \Rightarrow Weapon(x)$
  - $Enemy(x, America) \Rightarrow Hostile(x)$
  - $American(West)$
  - $Enemy(Nano, America)$

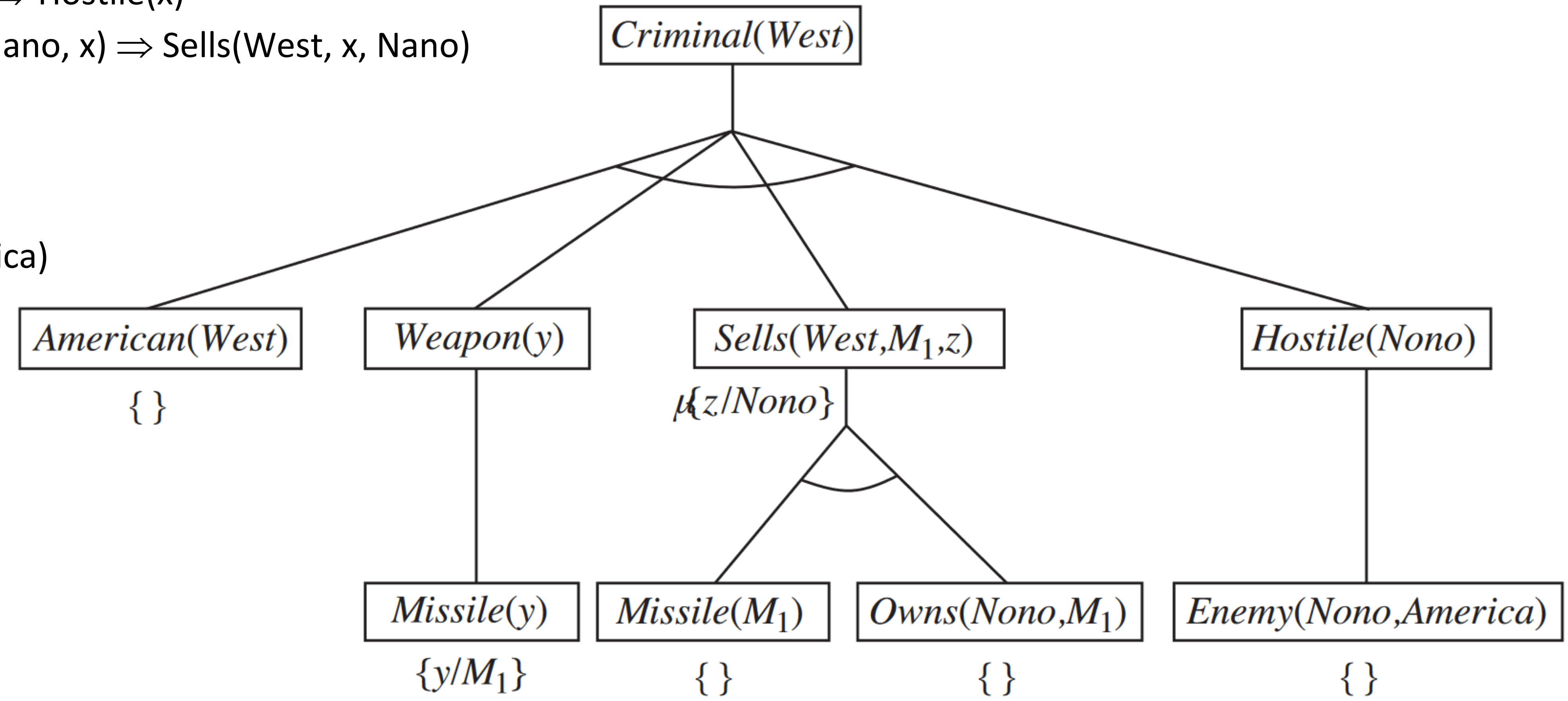
Forward Chaining

- First-Order Definite Clauses
  - American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
  - Missile(x) ⇒ Weapon(x)
  - Enemy(x, America) ⇒ Hostile(x)
  - Missile(x) ∧ Owns(Nano, x) ⇒ Sells(West, x, Nano)
  - Owns(Nono,M1)
  - Missile(M1)
  - American(West)
  - Enemy(Nano, America)



Backward Chaining

- First-Order Definite Clauses
  - American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
  - Missile(x) ⇒ Weapon(x)
  - Enemy(x, America) ⇒ Hostile(x)
  - Missile(x) ∧ Owns(Nano, x) ⇒ Sells(West, x, Nano)
  - Owns(Nono,M1)
  - Missile(M1)
  - American(West)
  - Enemy(Nano, America)



Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power: sufficient to express real-world problems
- Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses

- Problems:
  - Handling human conceptual categories, uncertainty and dynamics
- Next week: Planning with FOL