INF5390 – Kunstig intelligens

First-Order Logic

Roar Fjellheim

Outline

- Logical commitments
- First-order logic
- First-order inference
- Resolution rule
- Reasoning systems
- Summary

Extracts from AIMA
    Chapter 8: First-Order Logic
    Chapter 9: Inference in First-Order Logic

From propositonal to first-order logic

- Features of propositional logic:
  - + Declarative
  - + Compositional
  - + ”Possible-world” semantics
  - - Lacks expressiveness
- *First-order logic*:
  - √ Extends propositional logic
  - √ Keeps good features
  - √ Adds expressiveness

First-order logic

- First-order logic is based on “common sense” or linguistic concepts:
  - √ *Objects*: people, houses, numbers, ..
  - √ *Properties*: tall, red, ..
  - √ *Relations*: brother, bigger than, ..
  - √ *Functions*: father of, roof of, ..
  - √ *Variables*: x, y, .. (takes objects as values)
- First-order logic is the most important and best understood logic in philosophy, mathematics, and AI

Logic “commitments”

- *Ontological* commitment
  - Ontology - in philosophy, the study of “what is”
  - What are the underlying assumptions of the logic with respect to the nature of *reality*
- *Epistemological* commitment
  - Epistemology - in philosophy, the study of “what can be known”
  - What are the underlying assumptions of the logic with respect to the nature of what the agent can *know*

Commitments of some logic languages

Language	**Ontological Commitment**	**Epistemological Commitment**
Propositional logic	Facts	True/false/unknown
First-order logic	Facts, objects, relations	True/false/unknown
Temporal logic	Facts, objects, relations, times	True/false/unknown
Probability theory	Facts	Degree of belief 0 .. 1
Fuzzy logic	Facts w/degree of truth	Known interval value

First-order logic - syntax

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \\
&\quad\;\Big|\; \mathit{Sentence}\ \mathit{Connective}\ \mathit{Sentence} \\
&\quad\;\Big|\; \mathit{Quantifier}\ \mathit{Variable}, \Lambda\ \ \mathit{Sentence} \\
&\quad\;\Big|\; \neg \mathit{Sentence} \\
&\quad\;\Big|\; (\mathit{Sentence}) \\
\mathit{AtomicSentence} &\rightarrow \mathrm{Pr}\,\mathit{edicate}(\mathit{Term}, \mathrm{K}\ ) \mid \mathit{Term} = \mathit{Term} \\
\mathit{Term} &\rightarrow \mathit{Function}(\mathit{Term}, \Lambda\ ) \mid \mathit{Cons}\,\mathrm{tan}\,t \mid \mathit{Variable} \\
\mathit{Connective} &\rightarrow \wedge \mid \vee \mid \Leftrightarrow \mid \Rightarrow \\
\mathit{Quantifier} &\rightarrow \forall \mid \exists
\end{aligned}
$$

Constants, predicates, functions and terms

- *Constant* symbols
  - Refer to specific objects in the world: *John, 3, ..*
- *Predicate* symbols
  - Refer to particular relations in the model: *Brother, LargerThan*, i.e. sets of objects that satisfy the relation
- *Function* symbols
  - Refer to many-to-one object mappings: *FatherOf*
- *Term*, a logical expression
  - Refers to an object in the model
  - Can be a Constant, a Variable, or a Function of other terms

Atomic and complex sentences

- *Atomic* sentences
  - State basic facts
  - Examples: *Brother(Richard, John)*
    Married(FatherOf(Richard),MotherOf(John))
- *Complex* sentences
  - Composed of atomic sentences by using logical connectives (same as for propositional logic)
  - Examples: *Brother(Richard, John)*$\wedge$*Brother(John, Richard)*
    Older(John, 30) => $\neg$ Younger(John, 30)

Quantifiers and equality

- Quantifiers are used to express properties of *classes of objects*
- *Universal* quantification $\forall$
  - "For all .." quantification
  - Example: $\forall x\ Cat(x) \Rightarrow Mammal(x)$
- *Existential* quantification $\exists$
  - "There exists .." quantification
  - Example: $\exists x\ Sister(x, Spot) \wedge Cat(x)$
- Equality =
  - Two terms refer to same object
  - Example: *Father(John) = Henry*

Using first-order logic: Kinship domain

- The *domain* of family relationship
  - Objects: *Persons*
  - Unary predicates: *Male*, *Female*
  - Binary predicates: *Parent*, *Sibling, Brother*, *Sister*, …
  - Functions: *Mother*, *Father*
- Example sentences

$$\forall m,c\ Mother(c)=m \Leftrightarrow Female(m) \wedge Parent(m,c)$$

$$\forall w,h\ Husband(h,w) \Leftrightarrow Male(h) \wedge Spouse(h,w)$$

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

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

TELLing and ASKing sentences

- TELL an agent about kinship using *assertions*
  - TELL(KB, ($\forall m,c \; Mother(c) = m \Leftrightarrow Female(m) \wedge Parent(m,c)$))
  - TELL(KB, ($Female(Maxi) \wedge Parent(Maxi, Spot) \wedge Parent(Spot, Boots)$))
- ASK an agent by posing *queries* (or *goals*)
  - ASK(*KB*,($Grandparent(Maxi, Boots)$))
    - Answer: *Yes*
  - ASK(*KB*, ($\exists x \; Child(x, Spot)$))
    - Answer: $x = Boots$

Wumpus world revisited

- *Properties* of locations

  $\forall x, t\ \ At(Agent,x,t) \wedge Breeze(t) \Rightarrow Breezy(x)$

- *Diagnostic* rules – infer cause from effect

  $\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \wedge Adjacent(x,y)$

- *Causal* rules – infer effect from cause

  $\forall x,y\ Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(y)$

- *Definition* of predicate

  $\forall y\ Breezy(y) \Leftrightarrow (\exists x\ Pit(x) \wedge Adjacent(x,y))$

Inference in first-order logic

- Reduce to propositional inference
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

- Require *substitution* and/or *unification* of terms

Substitution and unification

- *Substitution* is the replacement of variable(s) in a sentence with expressions
  - √ SUBST({*x*/*Richard*, *y*/*John*}, *Brother*(*x*,*y*)) = *Brother*(*Richard*, *John*)
- *Unification* is finding substitutions (a *unifier*) that make different sentences look identical
  - √ UNIFY(*p*,*q*) = *Θ* where SUBST(*Θ*,*p*) = SUBST(*Θ*,*p*)
  - √ E.g.

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

$$UNIFY(Knows(John,x),Knows(y/Bill)) = \{x/Bill, y/John\}$$

$$UNIFY(Knows(John,x),Knows(y,Mother(y))) = \{y/John, x/Mother(John)\}$$

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

Example knowledge base

- Known facts
  - √ It is a crime for an American to sell weapons to hostile nations. The country Nono is an enemy of America and has some missiles, all sold to it by Colonel West, an America.
- Knowledge base:

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

$$Owns(Nono, M1), Missile(M1), Missile(x) \Rightarrow Weapon(x)$$

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

$$Enemy(x, America) \Rightarrow Hostile(x)$$

$$American(West), Enemy(Nono, America)$$

- Prove that West is a criminal

Reduce to propositional inference

- Can reduce first-order sentence to propositional sentence by instantiation rules
  - √ *UI - Universal instantiation* - Substitutes a constant in KB for a universally quantified variable
  - √ *EI - Existential instantiation* - Substitutes a *new* constant for an existentially quantified variable
- Apply UI and EI systematically to replace all sentences with quantifiers with variable-free sentences
- Can the propositional inference rules to derive proofs, but it is an inefficient procedure

Generalized modus ponens - GMP

- Assumes knowledge base containing facts and rules like

$$p_1 \wedge p_2 \wedge \Lambda \wedge p_n \Rightarrow q$$

- The *generalized modus ponens* rule

$$\frac{p'_1, p'_2, \Lambda\ , p'_n, (p_1 \wedge p_2 \wedge \Lambda \wedge p_n \Rightarrow q)}{SUBST(\theta, q)} \qquad \begin{array}{l} SUBST(\theta, p'_i) = \\ SUBST(\theta, p_i) \quad for\ all\ i \end{array}$$

- Example

$$SUBST = \{x / M1\}$$

$$Missile(M1)$$

$$Owns(Nono, M1)$$

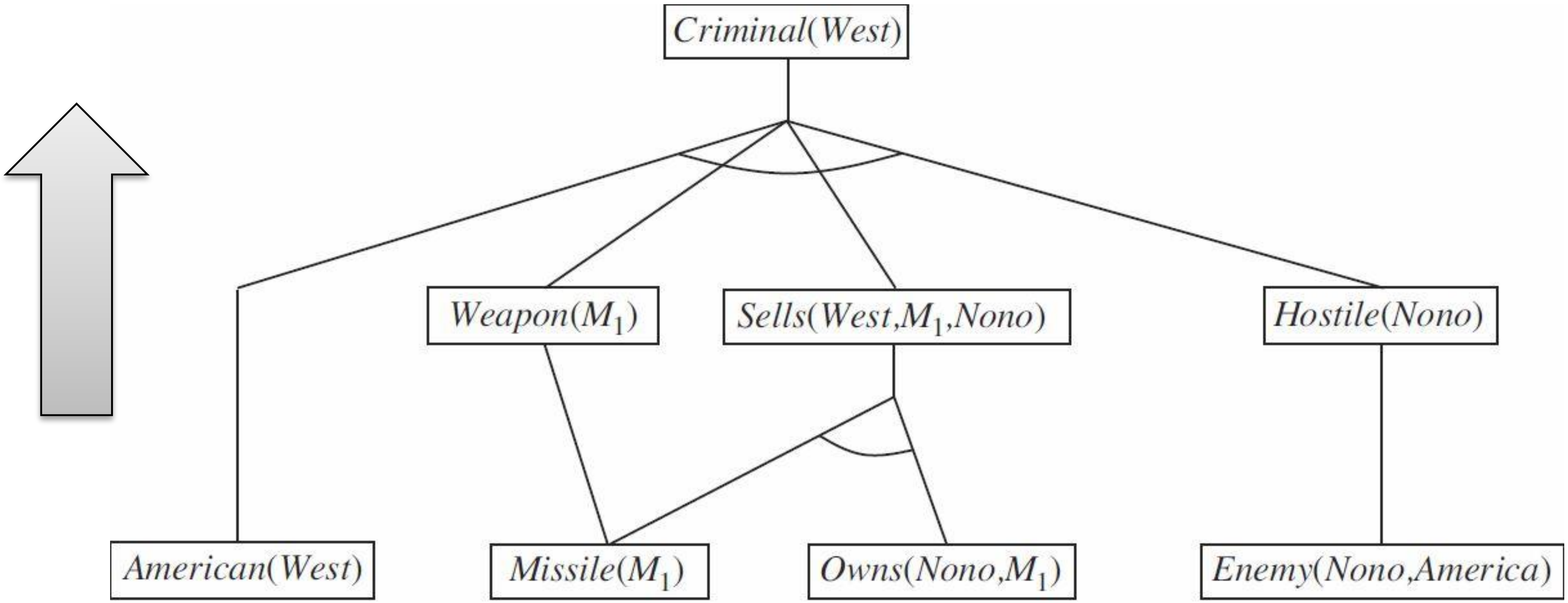
$$\frac{\forall x\ Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, Nono, x)}{Sells(West, Nono, M1)}$$

Completeness of first-order logic

- First-order logic is *complete*
  - If a sentence is entailed by KB, then this can be proved
  - (Gödel’s completeness theorem, 1930)
  - But generalized modus ponens is *not* complete
- However, first-order logic is *semi-decidable*
  - If the sentence is *not* entailed by the KB, this can not always be shown
- Extension of first-order logic with mathematical induction is incomplete
  - There are true statements that cannot be proved
  - (Gödel’s incompleteness theorem, 1931)

Forward chaining - FC

- Start with sentences in KB, apply inference rules in forward direction, adding new sentences until goal found or no further inference can be made

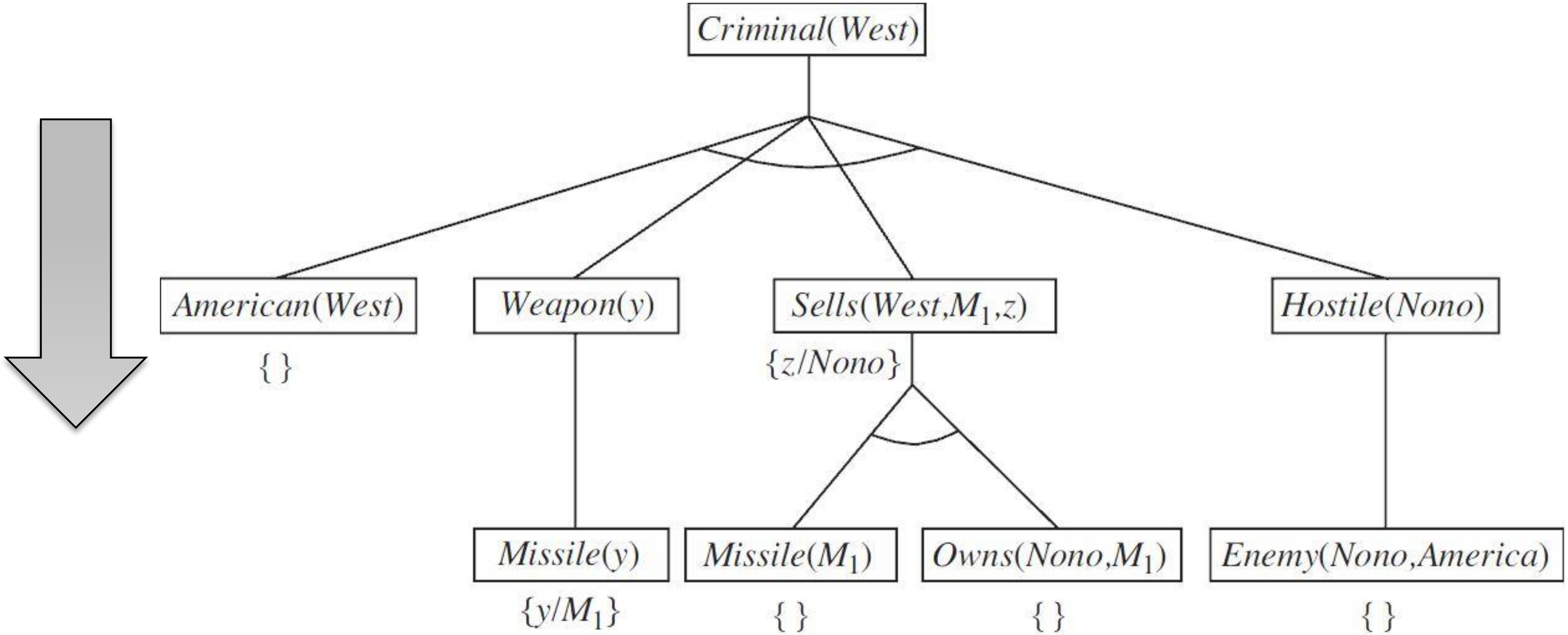


Production systems using FC

- Main features
  - Consists of rule base and working memory
  - Uses rule matching and forward chaining to add facts to the working memory until a solution is found
  - *Rete networks* are used to speed up matching
- Applications
  - Used in many expert systems, especially early ones
  - Design and configuration systems
  - Real-time monitoring and alarming
  - “Cognitive architectures” (SOAR)

Backward chaining - BC

- Start with goal sentence, search for rules that support goal, adding new sub-goals until match with KB facts or no further inference can be made



Logic programming using BC

- Main features
  - Restricted form of first-order logic
  - Mixes control information with declarative sentences
  - Backward chaining search: Prove a goal
- *Prolog* is the dominant logic programming language, and has been used for
  - Expert systems
  - Natural language systems
  - Compilers
  - Many others

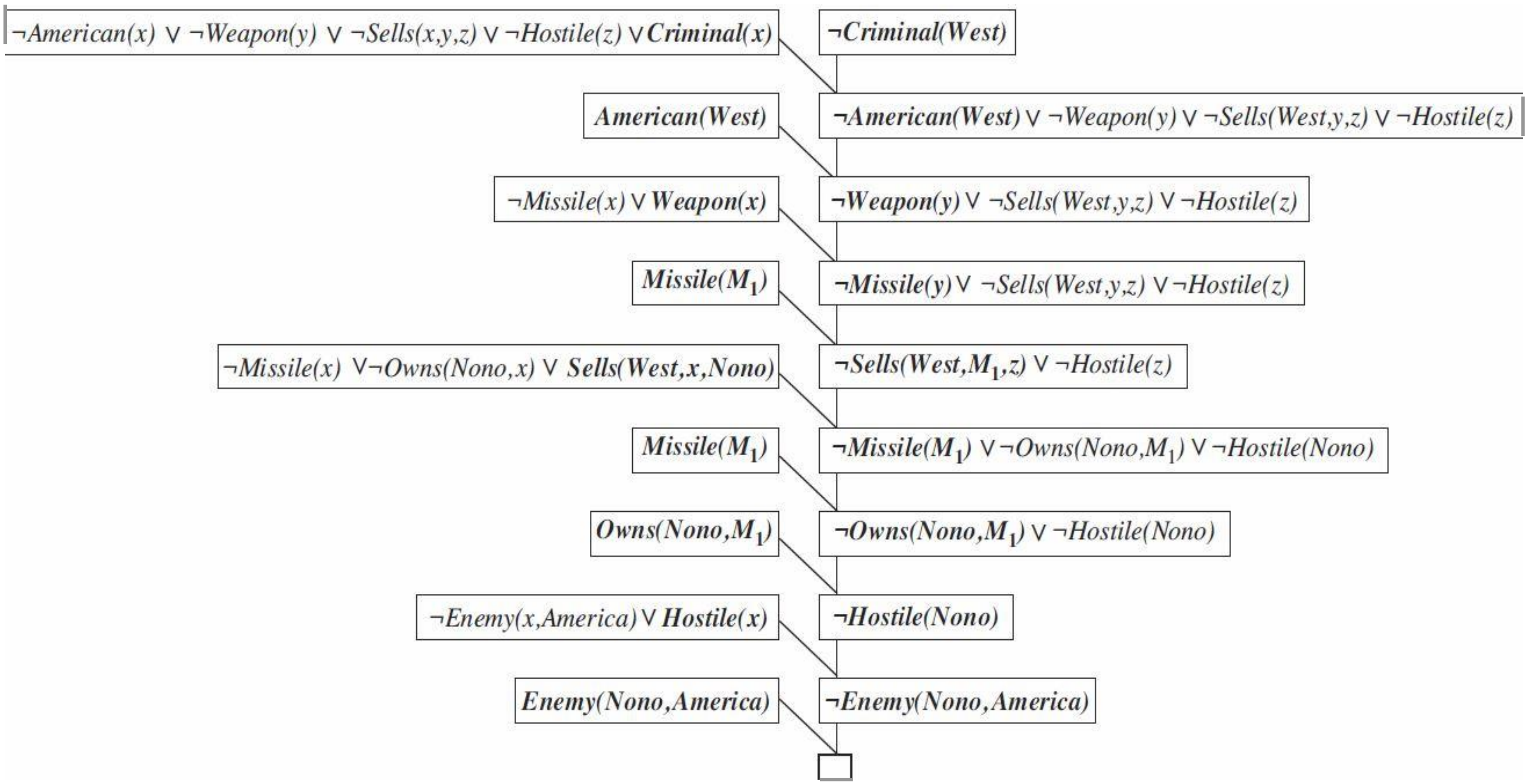
Resolution - A complete inference procedure

- Extends generalized modus ponens by allowing rules with disjunctive conclusions

$$p_1 \wedge p_2 \wedge \Lambda \wedge p_n \Rightarrow q_1 \vee q_2 \vee \Lambda \vee q_m$$

- Resolution rule: $\dfrac{p \vee q, \neg p \vee r}{q \vee r}$
- Assumes that all sentences are written in *a normal form*, for which there exists an efficient algorithm
  - E.g. eliminate implications, replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$
- A complete resolution inference procedure
  - Works by *refutation*, i.e. include $\neg$*Goal* in the KB
  - Then apply resolution rule until a contradiction is found

Example resolution proof



Theorem provers using resolution

- Main features
  - Allows expression of problems in full first-order logic
  - May permit additional control information
  - Performs inference by resolution or other procedure
- Applications
  - Assistants for mathematicians
  - Synthesis and verification of hardware/software systems
  - Diagnosis of technical systems

Summary

- Different logics make different *commitments* about the world and what can be known
- *First-order logic* commits to the existence of objects and relations between objects
- An agent can use first-order logic for reasoning about world *states* and for deciding on *actions*
- First-order sentences are made of *terms*, *predicates*, *functions*, *quantifiers* and *connectives*
- First-order logic is *complete* but semi-decidable, but extensions are incomplete

Summary (cont.)

- Reduce to propositional inference by replacing variables with constants - Inefficient
- Generalized Modus Ponens (GMP) relies on *unification*, i.e. partial instantiation of variables
- *Forward chaining* uses GMP to derive new facts from known facts
- *Backward chaining* uses GMP to prove goals from known facts
- *Resolution* is a complete and efficient proof system for first-order logic
- Reasoning systems include *theorem provers*, *logic programming*, and *production systems*