FO

Logic, Management & Frames

Dr. Collin F. Lynch

AI Academy: North Carolina State University

AI Academy NC STATE UNIVERSITY

Agenda

First-Order Logic

Chaining

TMS

Knowledge Engineering

Prop variable = states of the world

X ≡ I am hungry
it's raining

First-Order Logic

Reifying objects.

First Order Logic

True ServerInst

- Set K of constant symbols: a, b, c, tweety, john, fido, ...
- Set V of variable symbols: T, U, V, W, X, Y, ...
- Set F of function symbols: F(), G(), H(), FatherOf(), FirstCat(), ...

∞

- Set P of predicate symbols: p(), q(), r(), likes(), red(), ... T/F

PrvsBu(X)

- Set of logical symbols: $\wedge$, $\vee$, $\Rightarrow$ $\Leftrightarrow$, $\neg$
- Various punctuation: ( ), [ ], etc.
- Equality for variables $Father(john) = henry$
- Universal quantification $\forall X$
- Existential quantification $\exists X$

Scope

Example

predicate.

constant

$$1.\ has - feathers(tweety)$$

$$2.\ \forall X(has - feathers(X) \Rightarrow bird(X))$$

$$3.\ bird(tweety)$$

Models to Interpretations

- A set of objects U, called the universe of discourse
- A function $I_K : K \rightarrow U$ (i.e., for each constant symbol k in K, an element u in U)
- For each n-ary function symbol f in F, a function $I_f : U_n \rightarrow U$ (i.e., for each set of terms used as function arguments, a term interpreted as the return value)
- For each n-ary predicate symbol p in P, a function $I_p : P \rightarrow P * (Ux...(ntimes)...xU)$
- (P* is the powerset, the set of all subsets.)
- Alternatively, $I(p) \in U_n$ is defined by $I(p) = \{(t_1, ..., t_n)|p(t_1, ..., t_n) = True\}$, those sets of terms which make the predicate true.

Big Ideas

1. Convert to a logic problem (Propositionalization).
2. Handle Possible mappings (Unification).
3. Search (FC, BC & Resolution).

Example 1

1 2

Assume that Tweety has feathers. Assume that everything that has feathers is a bird. Prove that Tweety is a bird.

3

- *Constants:* tweety
- *Variables:* X
- *Functions:* none
- *Predicates:*
  - $has - feathers/1$
  - $bird/1$

$$
\begin{array}{l}
1. has - feathers(tweety) \\
2. \forall X(has - feathers(X) \Rightarrow bird(X)) \\
\hline
3. bird(tweety)
\end{array}
$$

Universal Instantiation

We can replace any universally instantiated variable in a statement with any *ground term* (g).

$$\frac{\forall v \alpha}{subst(\{v/g\}, \alpha)} \tag{1}$$

$$\frac{\forall x : cat(X) \Rightarrow stupid(X) \land hairy(X)}{\begin{array}{l} cat(tom) \Rightarrow stupid(tom) \land hairy(tom) \\ cat(X_{4214}) \Rightarrow stupid(X_{4214}) \land hairy(X_{4214}) \\ cat(EnemyOf(Dog)) \Rightarrow stupid(EnemyOf(Dog)) \land hairy(EnemyOf(Dog)) \end{array}}$$

·tom

Existential Instantiation (Skolem 1)

We replace existential variables with a *new constant*.

$$\frac{\exists v \alpha}{subst(\{v/C_{2121}\}, \alpha)} \tag{2}$$

$$\frac{\exists x : cat(x) \wedge messy(x)}{\begin{array}{l} cat(C_{3311}) \wedge messy(C_{3311}) \\ cat(C_{344354}) \wedge messy(C_{344354}) \end{array}}$$

Generalized Modus Ponens (Lifted)

$$
\begin{array}{l}
1. cat(sylvester) \\
2. bird(tweety) \\
3. cat(a) \wedge bird(b) \Rightarrow getsBeatenUpBy(a, b, t) \\
4. \theta = \{a/sylvester, b/tweety\} \\
\hline
5. getsBeatenUpBy(sylvester, tweety, t)
\end{array}
$$

$$Unify(\alpha, \beta) = \theta) \quad \Rightarrow Subst(\theta, \alpha) = Subst(\theta, \beta)$$

Subsumption Lattice

"aabCCDx"
"aabQQZx"

$$\begin{tikzcd}
 & IsProcess(A) \wedge InGroup(B) \arrow[dl] \arrow[dr] & \\
IsProcess(20391) \wedge InGroup(B) \arrow[dr] & & IsProcess(A) \wedge InGroup(userspace) \arrow[dl] \\
 & IsProcess(20391) \wedge InGroup(userspace) &
\end{tikzcd}$$

Resolution

$$\frac{\alpha \quad \beta}{\gamma}$$

$$\frac{(\cancel{a} \vee \neg b \vee c) \quad (\cancel{\neg a} \vee g \vee d)}{(\neg b \vee c \vee g \vee d)}$$

Chaining

$KB \oplus \varphi \rightarrow$

Forward Chaining

1. $bird(opus)$
2. $bird(tweety)$
3. $yellow(tweety)$
4. $penguin(opus)$
5. $cat(sylvester)$
6. $cat(X) \Rightarrow hatesSnow(X)$
7. $penguin(Y) \Rightarrow lovesSnow(Y)$
8. $bird(Q) \Rightarrow flies(Q)$
9. $(bird(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$

KB

$?chases(sylvester, tweety)$

q

Forward Chaining

1.	$bird(opus)$
2.	$bird(tweety)$
3.	$yellow(tweety)$
4.	$penguin(opus)$
5.	$cat(sylvester)$
6.	$cat(X) \Rightarrow hatesSnow(X)$
7.	$penguin(Y) \Rightarrow lovesSnow(Y)$
8.	$bird(Q) \Rightarrow flies(Q)$
9.	$(flies(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$
$(1 + 8)$	$\theta = \{Q/opus\} : flies(opus)$
$(2 + 8)$	$\theta = \{Q/tweety\} : flies(tweety)$
$(4 + 7)$	$\theta = \{Y/opus\} : lovesSnow(opus)$
$(5 + 6)$	$\theta = \{X/sylvester\} : hatesSnow(sylvester)$

Forward Chaining

Sound
complete.

irrelevant

...

7. $penguin(Y) \Rightarrow lovesSnow(Y)$

8. $bird(Q) \Rightarrow flies(Q)$

9. $(flies(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$

---

10. $(1 + 8)\theta = \{Q/opus\} : flies(opus)$

11. $(2 + 8)\theta = \{Q/tweety\} : flies(tweety)$

12. $(4 + 7)\theta = \{Y/opus\} : lovesSnow(opus)$

13. $(5 + 6)\theta = \{X/sylvester\} : hatesSnow(sylvester)$

14. $(10 + 3 + 5 + 9)\theta = \{Z/tweety, Q/sylvester\}$
$chases(sylvester, tweety)$

Heuristics

- *Minimum Remaining Values:* Fewest of X
- *Incremental FC*: Add one thing at a time.
- *Goal Filtering*: Filter irrelevant objects.
- Always NP-Hard.

Backward Chaining

1. $bird(opus)$
2. $bird(tweety)$
3. $yellow(tweety)$
4. $penguin(opus)$
5. $cat(sylvester)$
6. $cat(X) \Rightarrow hatesSnow(X)$
7. $penguin(Y) \Rightarrow lovesSnow(Y)$
8. $bird(Q) \Rightarrow flies(Q)$
9. $(Flies(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$

? $(9)\theta = \{Z/sylvester, Q/opus\}$

$chases(opus, sylvester)$

? $cat(opus)$?

Backward Chaining

$\ldots$

6.	$cat(X) \Rightarrow hatesSnow(X)$
7.	$penguin(Y) \Rightarrow lovesSnow(Y)$
8.	$bird(Q) \Rightarrow flies(Q)$
9.	$(flies(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$
?	$(9)\theta = \{Z/opus, Q/sylvester\}$
	$chases(sylvester, opus)?$
?	$cat(sylvester)?$
14.	$(5)cat(sylvester)$
?	$yellow(opus)?$

$\ldots$

Backward Chaining

θ_0 q_0 q_1 q_2 0 1 2 3 5

Goal directed

$\ldots$

6. $cat(X) \Rightarrow hatesSnow(X)$
7. $penguin(Y) \Rightarrow lovesSnow(Y)$
8. $bird(Q) \Rightarrow fies(Q)$
9. $(flies(Z) \wedge yellow(Z) \wedge cat(Q)) \Rightarrow chases(Q, Z)$

? $(9)\theta = \{Z/tweety, Q/sylvester\}$
$chases(sylvester, tweety)$?

? $cat(sylvester)$?

19. $(5)cat(sylvester)$

? $yellow(tweety)$?

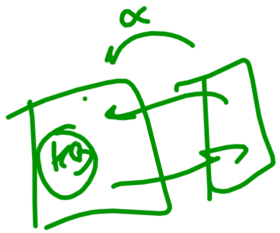
Wasted effort.

21. $(3)yellow(tweety)$

? $bird(tweety)$?

23. $(2)bird(tweety)\ldots$

Inferred chains.



Truth Maintenance

TMS

non-monotonic

Truth Maintenance Systems

Transactions.

- Allow KB updates with:
  - `Tell(KB,` α`)`: explicit addition of knowledge.
  - `Retract(KB,` α`)`: explicit retraction of knowledge.
- On each update we reevaluate the KB for potential changes.
- *OR* we store the **justification** with each entry:
  - We know Q because we know P and we know P implies Q.
  - $\{P,\ P \rightarrow Q\}\ :\ \ Q$
- We can also operationalize **assumptions** with entries.
- And we can generate **explanations**.

TMS Operation

1 Tell a

2 Tell b

3 Tell $b \wedge a \to g$

4 Tell $z \to g$

Tell z

retract a

~~{Told a}~~

{Told b}

{Told $b \wedge a \to g$} ~~{(b, a, $b \wedge a \to g$), g}~~

1, 2, 3

{Told $z \to g$}

{Tell z} ~~{z, $z \to g$, g}~~

{(~~b, a, $b \wedge a \to g$~~) (z, $z \to g$) g}

{ a : Told

g : [(a, b, $b \wedge a \to g$) . . .] }

Knowledge Engineering

Designing Software

Model Software

What is ~~Knowledge~~ Engineering?

Planning
Design
Setting requirements
Implementation
Testing
Debugging

Problem Solving

- Real-World Problem Solving
  - Diagnose a patient.
  - Design a bridge.
  - Calculate your taxes.
- Problem Solving
  - *Refine* the problem to make it solvable.
  - *Implement* a solution.
  - *Analyze* the result.

Knowledge Engineering

- Knowledge Engineering
  - Building knowledge Based Systems
  - Designing knowledge bases.
  - Developing task analyses. → How is this done?
  - Curating information.
  - Structuring data.
- Closely related to:
  - Systems analysis.
  - Task analysis.
  - Cognitive Task Analysis

Knowledge Representation

- Capturing (human) knowledge in a form the computer can reason about.
- General CS Problem:
  - General solutions are *data structures* (e.g. Lists)
  - AI solutions are *knowledge structures* (e.g. Ontologies)
- Well-designed KR should:
  - Represent all the knowledge that is *important* to the problem.
  - Reflect the structure of the domain at the right *granularity*.
  - Support incremental development.
- Well-designed KR should *not*:
  - Be *difficult* (inefficient) to construct or use.
  - Require excess knowledge.

Design problem: Modeling the "right" conditions, constraints, facts, etc. at the "right" level of detail in the "usable" way is hard!

Alternative Representations

- Logic & Predicate Calculus
- Rules & Production Systems if-then
- Description Logics, Semantic Networks, Frames
- Scripts
- Ontologies

Alternative Representations

- Logic & Predicate Calculus
  - Very very rich representation (sound and complete).
  - Does not always match humans *as we think*.
  - Interface can be inefficient.

Blackboard

FC

- Rules & Production Systems
  - Represent knowledge as if-then rules.
  - Inference can be quite efficient, natural, and modular.
  - Can even be non-monotonic.
  - Not necessarily sound or complete.
  - Conflict resolution required.
  - Inconsistencies create unintended consequences.

Structured, Framed, & Associative Representations

- Represent the structure of the domain then reason on that
  - Belief networks
  - Frames
  - Scripts
  - *Associative networks*
- Basic structure:
  - *Concepts* (nodes)
  - *Relationships* between concepts (is-a, part, etc.).
  - Can also include: Inheritance, Procedural attachment.