First Order Logic

Beyond Propositional logic

- Propositional logic not expressive enough
  - In Wumpus world we needed to explicitly write every case of Breeze & Pit relation
  - Facts = propositions
  - “All squares next to pits are breezy”
- “Regular” programming languages mix facts (data) and procedures (algorithms)
  - World[2,2]=Pit
  - Cannot deduce/compose facts automatically
  - Declarative vs. Procedural

Natural Language

- Natural language probably not used for representation
  - Used for communication
  - “Look!”

First-Order Logic

- Idea:
  - Don’t treat propositions as “atomic” entities.
- First-Order Logic:
  - Objects: cs4701, fred, ph219, emptylist …
  - Relations/Predicates: is_Man(fred), Located(cs4701, ph219), is_kind_of(apple, fruit)…
    - Note: Relations typically correspond to verbs
  - Functions: Best_friend(), beginning_of() : Returns object(s)
  - Connectives: $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
  - Quantifiers:
    - Universal: $\forall$x: (is_Man(x)) is_Mortal(x))
    - Existential: $\exists$y: (is_Father(y, fred))

Predicates

- In traditional grammar, a **predicate** is one of the two main parts of a sentence the other being the subject, which the predicate modifies.
- "John is yellow" *John* acts as the subject, and *is yellow* acts as the predicate.
- The predicate is much like a verb phrase.
- In linguistic semantics a **predicate** is an expression that can be *true of* something

Wikipedia

Types of formal mathematical logic

- Propositional logic
  - Propositions are interpreted as true or false
  - Infer truth of new propositions
- First order logic
  - Contains predicates, quantifiers and variables
    - E.g. Philosopher(a) $\rightarrow$ Scholar(a)
    - $\forall x$, King(x) $\wedge$ Greedy (x) $\Rightarrow$ Evil (x)
  - Variables range over individuals (domain of discourse)
- Second order logic
  - Quantify over predicates and over sets of variables

Other logics

- Temporal logic
  - Truths and relationships change and depend on time
- Fuzzy logic
  - Uncertainty, contradictions

Wumpus

- Squares neighboring the wumpus are smelly
  - Objects: Wumpus, squares
  - Property: Smelly
  - Relation: neighboring
- Evil king john rules England in 1200
  - Objects: John, England, 1200
  - Property: evil, king
  - Relation: ruled

Example: Representing Facts in First-Order Logic

1. Lucy* is a professor
2. All professors are people.
3. John is the dean.
4. Deans are professors.
5. All professors consider the dean a friend or don't know him.
6. Everyone is a friend of someone.
7. People only criticize people that are not their friends.
8. Lucy criticized John .

* Name changed for privacy reasons.

Same example, more formally

Knowledge base:

- is-prof(lucy)
- $\forall$ x (is-prof(x) $\rightarrow$ is-person(x))
- is-dean(John)
- $\forall$ x (is-dean(x) $\rightarrow$ is-prof(x))
- $\forall$ x ($\forall$ y (is-prof(x) $\wedge$ is-dean(y) $\rightarrow$ is-friend-of(y,x) $\vee$ $\neg$knows(x, y)))
- $\forall$ x ($\exists$ y (is-friend-of (y, x)))
- $\forall$ x ($\forall$ y (is-person(x) $\wedge$ is-person(y) $\wedge$ criticize (x,y) $\rightarrow$ $\neg$is-friend-of (y,x)))
- criticize(lucy, John)

Question: Is John no friend of Lucy?

$\neg$is-friend-of(John ,lucy)

How the machine “sees” it:

Knowledge base:

- $P1(A)$
- $\forall x\, (P1(x) \rightarrow P3(x))$
- $P4(B)$
- $\forall x\, (P4(x) \rightarrow P1(x))$
- $\forall x\, (\forall y\, (P1(x) \wedge P4(y) \rightarrow P2(y,x) \vee \neg P5(x, y)))$
- $\forall x\, (\exists y\, (P2(y, x)))$
- $\forall x\, (\forall y\, (P3(x) \wedge P3(y) \wedge P6(x,y) \rightarrow \neg P2(y,x)))$
- $P6(A, B)$

Question: $\neg P2(B, A)$?

Knowledge Engineering

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.

Knowledge Engineering

1. All professors are people.
2. Deans are professors.
3. All professors consider the dean a friend or don’t know him.
4. Everyone is a friend of someone.
5. People only criticize people that are not their friends.
6. Lucy* is a professor
7. John is the dean.
8. Lucy criticized John.
9. Is John a friend of Lucy’s?

General Knowledge

Specific problem

Query

Inference Procedures: Theoretical Results

- There exist complete and sound proof procedures for propositional and FOL.
  - Propositional logic
    - Use the definition of entailment directly. Proof procedure is exponential in *n*, the number of symbols.
    - In practice, can be much faster…
    - Polynomial-time inference procedure exists when KB is expressed as **Horn clauses**: $P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$

      where the P_i and Q are non-negated atoms.
  - First-Order logic
    - Godel's completeness theorem showed that a proof procedure exists…
    - But none was demonstrated until Robinson's 1965 *resolution algorithm*.
    - Entailment in first-order logic is *semidecidable*.

Types of inference

- Reduction to propositional logic
  - Then use propositional logic inference, e.g. enumeration, chaining
- Manipulate rules directly

Universal Instantiation

- $\forall x$, King(x) $\wedge$ Greedy (x) $\Rightarrow$ Evil (x)
  - King(John) $\wedge$ Greedy (John) $\Rightarrow$ Evil (John)
  - King(Richard) $\wedge$ Greedy (Richard) $\Rightarrow$ Evil (Richard)
  - King(Father(John)) $\wedge$ Greedy (Father(John)) $\Rightarrow$ Evil (Father(John))
- Enumerate all possibilities
  - All must be true

Existential Instantiation

- $\exists$ x, Crown(x) $\wedge$ OnHead(x, John)
  – Crown (C) $\wedge$ OnHead(C, John)
  – Provided C is not mentioned anywhere else
- Instantiate the one possibility
  – One must be true
  – Skolem Constant (skolemization)

Resolution Rule of Inference

Example:

Assume:	$E_1 \vee E_2$	playing tennis or raining
and	$\neg E_2 \vee E_3$	not raining or working
Then:	$E_1 \vee E_3$	playing tennis or working

"Resolvent"

General Rule:

Assume:	$E \vee E_{12} \vee ... \vee E_{1k}$
and	$\neg E \vee E_{22} \vee ... \vee E_{2l}$
Then:	$E_{12} \vee ... \vee E_{1k} \vee E_{22} \vee ... \vee E_{2l}$

Note: E_{ij} can be negated.

Algorithm: Resolution Proof

- Negate the original theorem to be proved, and add the result to the knowledge base.
- Bring knowledge base into conjunctive normal form (CNF)
  - CNF: conjunctions of disjunctions
  - Each disjunction is called a clause.
- Repeat until there is no resolvable pair of clauses:
  - Find resolvable clauses and resolve them.
  - Add the results of resolution to the knowledge base.
  - If NIL (empty clause) is produced, stop and report that the (original) theorem is true.
- Report that the (original) theorem is false.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg \alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg \alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg \alpha \vee \neg \beta) \quad \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg \alpha \wedge \neg \beta) \quad \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Resolution Example: Propositional Logic

- To prove: $\neg P$
- Transform Knowledge Base into CNF

	Regular	CNF
Sentence 1:	$P \rightarrow Q$	$\neg P \vee Q$
Sentence 2:	$Q \rightarrow R$	$\neg Q \vee R$
Sentence 3:	$\neg R$	$\neg R$

- Proof

1.	$\neg P \vee Q$	Sentence 1
2.	$\neg Q \vee R$	Sentence 2
3.	$\neg R$	Sentence 3
4.	*P*	Assume opposite
5.	*Q*	Resolve 4 and 1
6.	*R*	Resolve 5 and 2
7.	nil	Resolve 6 with 3

8. Therefore original theorem ($\neg P$) is true

Resolution Example: FOL

Axioms: Regular	CNF
$\forall x : feathers(x) \rightarrow bird(x)$	$\neg feathers(x) \vee bird(x)$
$feathers(tweety)$	$feathers(tweety)$

Is bird(tweety)?

A: True **B: False**

Resolution Example: FOL

Example: Prove *bird(tweety)*

Axioms:	Regular	CNF
1:	$\forall x : feathers(x) \rightarrow bird(x)$	$\neg feathers(x) \vee bird(x)$
2:	$feathers(tweety)$	$feathers(tweety)$
3:	$\neg bird(tweety)$	$\neg bird(tweety)$
4:		$\neg feathers(tweety)$

Resolution Proof

1. Resolve 3 and 1, specializing (i.e. “unifying”) tweety for x.
   Add :feathers(tweety)
2. Resolve 4 and 2. Add NIL.

Resolution Theorem Proving

Properties of Resolution Theorem Proving:

- sound (for propositional and FOL)
- (refutation) complete (for propositional and FOL)

Procedure may seem cumbersome but note that can be easily automated. Just “smash” clauses until empty clause or no more new clauses.

Pigeon-Hole (PH) problem: you cannot place n+1 pigeons in n holes (one per hole)

A note on negation

- To prove theorem θ we need to show it is never wrong:
  - we test if there is an instance that satisfies $\neg\theta$
  - if so report that θ is false
- But we are not proving that $\neg\theta$ is true
  - Just that θ is false
  - Showing instance of $\neg\theta$ is not the same as showing that $\neg\theta$ is *always* true
- E.g. prove theorem θ that says "$x+y=4 \rightarrow x=2 \wedge y=2$"
  - We find a case $x=1 \wedge y=3$ so theorem is not true
  - But $\neg\theta$ is also not always true either

Substitutions

- Syntax:
  – SUBST (A/B, q) or SUBST (θ, q)
- Meaning:
  – Replace All occurrences of “A” with “B” in expression “q”
- Rules for substitutions:
  – Can replace a variable by a constant.
  – Can replace a variable by a variable.
  – Can replace a variable by a function expression, as long as the function expression does not contain the variable.

$$v_1/C; v_2/v_3; v_4/f(\ldots)$$