First-Order Logic

Differences in Logics

- **Ontological Commitment** – What the logic assumes about the state of reality.
- **Epistemological Commitment –** the possible state of knowledge a logic allows with respect to each fact.

Language	Ontological	Epistemological
Propositional Logic	Facts	T/F/unknown
1st-order Logic	Facts, objects, relations	T/F/unknown
Temporal Logic	Facts, objects, relations, time	T/F/unknown
Probability Theory	Facts	Degree of belief
Fuzzy Logic	Facts with degree of truth	Known interval value

Syntax of First Order Logic

- **Objects**: the domain of the model is the set of objects in it ➔ *constant* symbols.
- **Relations**: a set of tuples of objects that are related ➔ *predicate* symbols.
- **Function**: an object can be related to exactly 1 object. Functions in FOL must be total functions that must be defined over the whole domain ➔ *function* symbols.
  - The *arity* of a relation or function is the number of arguments it takes.
- **intended interpretation** – the interpretation specify what each symbol actually represents in the real world.
- **term** – a logical expression that refers to an object.
  - **ground term** – a term with no variables.
- **atomic sentence** – a predicate symbol followed by a parenthesized list of terms; the arguments. An atomic sentence is true (under the given interpretation) if the predicate holds for the objects given as arguments.
- **complex sentence** – atomic sentences joined by logical connectives.
- **quantifiers** – allow us to express properties of groups of objects.
  - **Universal** $\forall x \quad P(x)$ - for all objects x, P(x) holds. P(x) usually formed with connective ==>.
  - **Existential** $\exists x \quad P(x)$ - there exists an object x, such that P(x) holds. Typically used with the connective AND.
- **Equality –** Asserts that two statements are equivalent.

Semantics of First-Order Logic

- **assertion** – sentences that assert what is known about the world.
- **queries (goals)** – questions asked to the KB. Answered in a list of substitutions that satisfy the query.
- **substitution (binding list)** $\{x/X\}$ - set of variable/term pairs that represent substituting the term *X* for the variable *x*.
    - $SUBST(\theta,\alpha)$ applies the substitution θ to sentence α.
- **axiom** – basic factual information from which conclusions are derived. Each axiom cannot be entailed by the KB without explicitly including it.
- **theorems** – facts entailed by the axioms.
- **perception –** the reasoning process by which percepts are entailed from the KB.

Types of rules in FOL

- **synchronic** – sentences relating properties of a world state to other properties in the same world state.
    - **Diagnostic Rules –** lead from observed effects to hidden causes.
    - **Casual Rules –** hidden properties cause observed precepts.
    - **Model-Based Reasoning –** Systems that use casual rules to model the world.
- **diachronic** - sentences relating properties of a world state to other properties in another world state; thus allowing reasoning across time.
- *If the axioms correctly and completely describe the way the world works and the way that percepts are produced, then any complete logical inference procedure will infer the strongest possible description of the world state given the available precepts.*

Knowledge Engineering – process of constructing a knowledge base.

1. Identify the task
2. Assemble relevant knowledge; **knowledge acquisition**
3. Decide on vocabulary of predicates, functions, and constants = **ontology**.
4. Encode general knowledge about the domain = **axioms**.
5. Encode a description of the specific problem instance = knowledge base.
6. Pose queries KB and get answers.
7. Debug the knowledge base.

Inference in First-Order Logic

*The question of entailment for first-order logic is **semidecidable** – algorithms exist that are able to correctly identify every entailed sentence, but no algorithm can correctly identify every nonentailed sentence.*

Fundamentals

- Converting quantified statements
  - Universal Instantiation – substitute variable of a universally quantified statement with a ground term. Can be applied many times producing several consequences.

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

  - Existential Instantiation – substituting a ground term, Skolem constant, for the variable in an existential statement. Can only be applied once after which the sentence is discarded.

$$\frac{\exists v \quad \alpha}{\text{SUBST}(\{v/k\}, \alpha)}$$

    If the Existential quantifier is embedded in a universally quantified statement, must use a skolem function; a generic function of the universally quantified variable.
- **Propositionalization** – the process of converting a first-order sentences into propositional logic.
  - a propositionalized KB is *inferentially equivalent* to the original KB but not *logically equivalent*.
  - entailment is preserved thus resolution is *complete*.
  - functions cause the number of possible ground terms to be infinite.
  - This process begins by converting the KB into CNF.
- **Conjunctive Normal Form** – a conjunction of clauses where each clause is a disjunction of literals.
  - *Every sentence in FOL can be converted into an inferentially equivalent CNF sentence.*
  - Conversion to CNF:
    1. Eliminate implications using: $p \Rightarrow q \quad \equiv \quad \neg p \vee q$
    2. Move $\neg$ inwards in quantified statements.
    3. **Standardize Variables**: eliminate repeated names.
    4. **Skolemization**: removal of existential quantifiers with Skolem functions (for every enclosing universal quantifier variable) or Skolem constants.
    5. Drop universal quantifiers.
    6. Distribute $\wedge$ over $\vee$.

- **Unification** – the process of finding substitutions that make different logical expressions look identical. Given two sentences, *p* and *q*, UNIFY returns the most general unifier θ, if a unifier exists.

$$\text{UNIFY}(p,q)=\theta \quad | \quad \text{SUBST}(\theta,p)=\text{SUBST}(\theta,q)$$

  - standardizing apart – renaming variables to avoid name clashes.
  - most general unifier – for every unifiable pair of expressions there is a unique unifier that is more general than any other.
    - A unifier is *more general* than another unifier if the first places fewer restrictions on the values of the variables.
    - A recursive algorithm for computing the most general unifier is given on pg. 278.
  - occur check – when matching a variable against a complex term, one must check whether the variable itself occurs in the term, in which case the unification fails.
- **Generalized Modus Ponens** – A lifted version of Modus Pones for FOL. For atomic sentences p_i, p_i', and q, where $\exists\theta\left(\forall i \quad \text{SUBST}(\theta,p_i')=\text{SUBST}(\theta,p_i)\right)$

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{SUBST(\theta, q)}$$

Forward Chaining – Uses Modus Pones to infer new atomic sentences in the KB until no more inferences are possible. This approach is *sound* and is *complete* for definite clause KBs. FOL Inference with definite clauses is *semidecidable* due to functions.

- *Datalog KB* – set of first-order definite clauses with no function symbols.
- *fixed point* – a state of the KB for which no further sentences can be inferred.
- Requirements for Efficient Forward Chaining.
  1. **Pattern Matching** – finding all possible unifiers for a rule.
     - *Any finite-domain CSP can be expressed as a single definite clause together with associated ground facts. Hence, matching a definite clause against a set of facts is NP-HARD in general.*
       - But, most rules in KB's are small and simple
     - *Data Complexity* – complexity of inference as a function of number of ground facts; is polynomial.
  2. Preventing Redundant Checking of rules
     - *Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration t-1.*
     - Thus, we retain and complete partial matches ➔ no recomputation.
     - The *rete algorithm* uses this process to make efficient inference by constructing a dataflow network where each node is a literal from a rule premise… this network captures all partial matches.
  3. Preventing Facts Irrelevant to the Goal.
     - Backward Chaining is an alternative.
     - Restrict forward chaining to a selected subset of rules.
     - Rewrite rule set using info from the goal, so that only relevant variable bindings, the *magic set*, are considered.

Backward Chaining – works backwards from the goal, matching the effects of rules to support the desired proof based on Modus Pones. It is a DFS search ➔ linear space complexity; repeated states and incompleteness.

- Algorithm outline:
  - A list of unsatisfied goals is maintained as a stack, 1 stack for each branch of the proof.
  - If all goals of the stack are satisfied by the initial state, the proof succeeds.
  - Goals are popped off the stack and if unsatisfied:
    - Every clause whose *head* unifies with the goal makes a separate branch of the proof.
    - The *body* of each such clause are added to the stack: *a new branch*.
- Problems with Backward Chaining
  1. *Repeated States* – inference can be exponential in number of ground facts.
     - **memoization** – caching solutions to subgoals and reusing the solutions when they reoccur.
  2. *Infinite Paths* – possibility of infinite paths makes backward chaining incomplete.
- **Logic Programming** – logic is expressed as formal declarative language and used to solve problems via inference.
  - *Prolog* – logic PL with depth-first backward chaining.
    - Rules expressed as follows:

      $$p :- n_1, \ldots, n_m. \quad \equiv \quad n_1 \wedge \ldots \wedge n_m \Rightarrow p$$

    - Built in arithmetic functions and some predicates have side-effects
    - **Negation as failure** – if a goal P can not be proved, $\neg P$ is considered true.
    - *Occur Check is omitted ➔ unsound.*
    - Prolog can be compiled or interpreted.
      - compiled – a miniature theorem prover is created.
      - interpreted – intermediate language execued by Warren Abstract Machine.
  - **choice point** – a structure for remembering previous choices in the DFS.
    - **continuation** – implementation of a choice point which contains a procedure call that defines what to do when goal succeeds.
  - **trail** – a list of bound variables used to unbind during backtracking.
  - **OR-parallelism** – possibility of a goal unifying with different clauses gives rise to solving each different branch in parallel.
  - **AND-parallelism** – possibility of solving each conjunct in the body of an implication in parallel. *Each conjunctive branch must communicate bindings made with other branches.*
- **Constraint Logic Programming** – CSP's can be encoded as definite clauses. Backward Chaining can only used for finite domain CSPs.
  - The idea is to allow variables to be constrained (restrict values) rather than bound (fixed to a single value)
  - MRS language – allows *metarules* to determine which conjuncts to try 1st.

Generalized Resolution – an extension of propositional resolution to FOL.

- **Proof by contradiction** – *The goal is negated and added to the KB. If the empty clause {} is derived, the goal has been proved.*
  - Proofs derived in this way are *non-constructive*: they only indicate whether the query is true or false, not what variables make it so.
    1. Restrict query variables to a single binding and backtrack.
    2. Add an **answer literal** as a disjunction with the negated goal. The resulting non-constructive proof will have a disjunction of possible answers instead of an empty clause → multiple answers.
- Algorithm outline
  - Begin with propositionalization and conversion to CNF. Each conjunction of clauses is broken into individual clauses.
  - The binary resolution rule can be applied to pairs of clauses (assumingly standardized apart) if they contain complementary literals.
  - The Resolution Inference Rule
    - *First-order literals are complementary if one unifies with the negation of the other.*
    - **binary resolution rule**:

$$\frac{l_1 \vee \ldots \vee l_k \quad m_1 \vee \ldots \vee m_n}{\text{SUBST}\left(\theta, l_1 \vee \ldots \vee l_{i-1} \vee l_{l+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n\right)}$$

$$\text{where } UNIFY\left(l_i, \neg m_j\right) = \theta$$

    - **factoring** – reduces two literals to one if they are unifiable.
  - Together, the *binary resolution rule* and *factoring* are complete.
- **Completeness of Resolution** – *If S is an unsatisfiable set of clauses, then the application of a finite number of resolution steps to S will yield a contradiction.*

Theorem Prover – an automated reasoner that differs from logic programming in that it accepts full first-order logic and the syntactic form chosen doesn't affect its results.

- Design of a Theorem Prover
  - **set of support** – a set of clauses that define the important facts.
  - **usable axioms** – background knowledge about the problem.
  - **demodulators (rewrites)** – set of equations applied in the left to right direction defining a canonical form for simplified terms.
  - A set of parameters and clauses defining control strategy.
- **proof checker** – proof given by a human is verified at each step by a Theorem prover.
- **Socratic reasoner** – a theorem prover with an incomplete ASK function, but which can always arrive at a solution if asked the right series of questions. This is a form of an assistant.
- **deductive synthesis** – create of a program *p* satisfying a set of specifications where the program is verified by an exists query and extracted from the constructive proof.