Inference in First-Order Logic

Michael Rose

In which we define effective procedures for answering questions posted in first order logic.

9.1 | Propositional vs. First Order Inference

First order inference can be done by converting the knowledge base to propositional logic and using propositional inference.

9.1.1 | Inference Rules for Quantifiers

The rule of **universal instantiation** says that we can infer any sentence obtained by substituting a ground term (one without variables) for the variable.

In the rule of **existential instantiation**, the variable is replaced by a single new constant symbol. Formally, for any sentence α, variable v, and constant k that does not appear elsewhere in the knowledge base, $\frac{\exists v \alpha}{\text{Sub(v/k,}\alpha\text{)}}$

An example would be math. Suppose we discover a new constant. Then we can name it something, but we can't name it π. In logic, the new name is called a **Skolem constant**. Existential Instantiation is a special case of a more general process called **skolemization**.

The question of entailment for first order logic is **semidecidable** – algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.

9.2 | Unification and Lifting

Generalized Modus Ponens: For atomic sentences p_i, p_i', q, there is a substitution θ such that $\text{subst}(\theta, p_i') = \text{subst}(\theta, p_i)$ for all i, $\frac{p_1', p_2', \ldots, (p_1 \wedge p_2 \wedge \ldots \wedge p_n \implies q)}{\text{subst}(\theta, q)}$

Generalized Modus Ponens is a lifted version of Modus Ponens, it raises it from ground (variable-free) propositional logic to first order logic.

9.2.2 | Unification

Lifted inference rules require finding substitutions that make different logical expressions look identical. This process is called **unification** and is a key component of all first order inference algorithms.

The process is simple: recursively explore the two expressions simultaneously side by side, building up a unifier along the way, but failing if two corresponding points in the structures do not match.

9.3 | Forward Chaining

9.3.3 | Efficient Forward Chaining

Suppose we wish to find answers to $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$

We could first find all the objects owned by Nono in constant time per object, then for each object, check whether it is a missile. If Nono owns many objects and very few missiles, it would be better to find all the missiles first and then check whether they are owned by Nono. This is the **Conjunct Ordering** problem: Find an ordering to solve the conjuncts of the rule premise so that the total cost is minimized. It turns out finding the optimal ordering is NP-hard, but good heuristics are available.

A heuristic example is the **minimum remaining values (MRV)** heuristic used for CSPs in ch 6, which suggests that we look for missiles first if fewer missiles than objects are owned by Nono.

9.4.4 | Redundant Inference and Infinite Loops

Forward chaining on graph search problems is an example of **dynamic programming**, in which the solutions to the subproblems are constructed incrementally from those of smaller subproblems and cached to avoid recomputation. We can obtain a similar effect in a backward chaining system using **memoization**.

9.4.6 | Constrain Logic Programming

CLP allows variables to be constrained rather than bound. A CLP solution is the most specific set of constraints on the query variables that can be derived from the knowledge base.

9.5 | Resolution

In the last chapter, we saw that propositional resolution using refutation is a complete inference procedure for propositional logic. This section extends resolution to first order logic.

9.5.1 | Conjunctive Normal Form for First Order Logic

As in the propositional case, first order resolution requires that sentences be in **conjunctive normal form**, that is, a conjunction of clauses, in which each clause is a disjunction of literals.

Every sentence of first order logic can be converted into an inferentially equivalent CNF sentence.

9.5.5 | Equality

None of the inference methods described so far handle equality, e.g. $x = y$.

We can use some inference rules to handle equality. The simplest is **demodulation**, which takes a unit clause $x = y$ and some clause α that contains the term x, and yields a new clause formed by substituting y for $x \in \alpha$.

As an example:

$Father(Father(x)) = PaternalGrandfather(x)\ Birthdate(Father(Father(Bella)), 1926)$

which concludes with the demodulation

$Birthdate(PaternalGrandfather(Bella), 1926)$.

We can also extend this rule to handle all non-unit clauses in which an equality literal appears, which is called **paramodulation**. For example, from $P(F(x, B), x) \lor Q(x)$ and $F(A, y) = y \lor R(y)$, we have $\theta = Unify(F(A, y), F(x, B)) = \{x/A, y/B\}$, and we conclude by paramodulation the sentence $P(B, A) \lor Q(A) \lor R(B)$.

Paramodulation yields a complete inference procedure for first order logic with equality.

There also exists extended unification algorithms which can show terms as unifiable if they are provably equal under some substitution.

9.5.6 | Resolution Strategies

Repeated applications of the resolution inference rule will eventually find a proof if one exists. Here are some strategies to find proofs efficiently.

Unit Preference - This strategy prefers to do resolutions where one of the sentences is a single literal (also known as a unit clause). **Unit Resolution** is a restricted form of resolution in which every resolution step must involve a unit clause.

Set of support - In general, it is effective to try to eliminate some potential resolutions altogether. We can insist that every resolution step involve at least one element of a special set of clauses - the set of support. The resolvent is then added to the set of support. If the set of support is small relative to the whole knowledge base, the search space will be reduced dramatically. The set of support strategy has the added benefit of generating goal directed proof trees that are often easy for humans to understand.

Input Resolution - In this strategy, every resolution combines one of the input sentences with some other sentence. The **linear resolution** strategy is a generalization in that allows P and Q to be resolved together either if P is in the original KB or if P is an ancestor of Q in the proof tree. Linear Resolution is complete.

Subsumption - This method eliminates all sentences that are subsumed by (that is, more specific than) an existing sentence in the KB.

9.6 | Summary

- A first approach uses inference rules (universal instantiation and existential instantiation) to propositionalize the inference problem. Typically, this approach is slow, unless the domain is small.
- The ease of unification to identify appropriate substitutions for variables eliminates the instantiation step in first order proofs, making the process more efficient in many cases.
- A lifted version of Modus Ponens uses unification to provide a natural and powerful inference rule, generalized Modus Ponens. The forward chaining and backward chaining algorithms apply this rule to sets of definite clauses.
- Generalized Modus Ponens is complete for definite clauses, although the entailment problem is semidecidable. For datalog knowledge bases consisting of function free definite clauses, entailment is decidable.

Forward chaining is used in deductive databases, where it can be combined with relation database operations. It is also used in production systems, which perform efficient updates with very large rule sets. Forward chaining is complete for datalog systems and runs in polynomial time.

- Backward chaining is used in logic programming systems, which employ sophisticated compiler technology to provide very fast inference. Backward chaining suffers from redundant inferences and infinite loops - these can be alleviated by memoization.
- Prolog, unlike first order logic, uses a closed world with the unique names assumption and negation as failure. These make Prolog a more practical programming language, but bring it further from pure logic.
- The generalized resolution inference rule provides a complete proof system for first order logic, using knowledge bases in conjunctive normal form.

- Several strategies exist for reducing the search space of a resolution system without compromising completeness. One of the most important issues is dealing with equality; demodulation and paramodulation can be used.
- Efficient resolution based theorem provers have been used to prove interesting mathematical theorems and to verify and synthesize software and hardware.