Artificial Intelligence

Inference in First Order Logic

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

  $\forall v\ \alpha \models \text{Substitute}(\{v/g\}, \alpha)$

  for any variable v and ground term g

Existential instantiation (EI)

- For any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base:
  $\exists v\ \alpha \models$ Substitute($\{v/k\}$, α)

- k is called Skolem constant

FOL Inference By Reduction

- Reduction to propositional logic
  - Instantiating the universal sentence in all possible ways
  - Give each ground term sentence a proposition symbol
- Every FOL KB can be propositionalized so as to preserve entailment
- A ground sentence is entailed by new KB iff entailed by original KB
- Idea: propositionalize KB and query, apply resolution, return result
- Problem:
  - with function symbols, there are infinitely many ground terms, such as Father(Father(Father(John)))
  - generate lots of irrelevant sentences

Reduction

- Theorem: If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB. (by Herbrand, 1930)

- Idea:
  For $n = 0$ to ∞ do
      create a propositional KB by instantiating with depth-n terms
      see if α is entailed by this KB

- Problem: works if α is entailed, loops if α is not entailed

- Theorem: Entailment for FOL is semi-decidable, that is, algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence. (by Turing, 1936, Church, 1936)

Unification

- A process of making two first order logic sentences with universal quantified variables identical by finding a substitution.

- Unify(α,β) = θ if αθ = βθ

- Transaction(Toys, x, John) & Transaction(Toys, Teddy, John), θ = {x/Teddy}}

- Transaction(Toys, x, y) & Transaction(z, Teddy, John), θ = {x/Teddy, y/John, z/Toys}}

- Transaction(Seller(x), x, John) & Transaction(y, Teddy, John), θ = {x/Teddy, y/ Seller(Teddy)}}

- Transaction(Toys, x, John) & Transaction(Toys, Teddy, x), θ = {fail}, but Standardizing apart eliminates overlap of variables, e.g., Transaction(Toys, Teddy, x_{17}) θ = {x/Teddy, x_{17}/John}

- Transaction(x, Teddy, x) & Transaction(Toys, Teddy, John) = {fail}

Most General Unifier

- Transaction(Toys, x, y) & Transaction(Toys, Teddy, z),
  θ = {x/Teddy, y/z }
  or θ = {x/Teddy, y/John, z/John}
  or θ = {x/Teddy, y/Mary, z/Mary}

- The first unifier is more general than the rest.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

- MGU = { x/Teddy, y/z }

The unification algorithm

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
    y, a variable, constant, list, or compound
    θ, the substitution built up so far

  if θ = failure **then return** failure
  else if $x = y$ **then return** θ
  else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) **and** COMPOUND?(y) **then**
    return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  else if LIST?(x) **and** LIST?(y) **then**
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure

The unification algorithm

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
    x, any expression
    θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Generalized Modus Ponens (GMP)

- $p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$

  where $p_i'\theta = p_i\theta$ for all i from 1 to n

- Example:
  Product(Teddy), Sells(Toys, Teddy),
  Transaction(Toys, Teddy, Mary),
  Transaction(x, y, z) $\Rightarrow$ Owns(z, y) $\models$ Owns(Mary, Teddy)

  θ is {x/Toys, y/Teddy, z/Mary}

- GMP can be used with KB of definite clauses (exactly one positive literal)

- All variables assumed universally quantified.
  (Existentially quantified variables are replaced by Skolem constants.)

Soundness of GMP

- Need to show that
  $p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$

  provided that $p_i'\theta = p_i\theta$ for all i from 1 to n

- Lemma: For any sentence p, we have $p \models p\theta$ by UI

- Proof:

  - $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta \models (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$

  - $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$

  - From previous two steps, $q\theta$ follows by ordinary Modus Ponens

Forward chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

    repeat until *new* is empty
        $new \leftarrow \{\}$
        for each sentence r **in** KB **do**
            $(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
            for each θ such that $(p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_n)\theta$
                    for some $p'_1, \ldots, p'_n$ in KB
                $q' \leftarrow$ SUBST(θ, q)
                if q' is not a renaming of a sentence already in KB or *new* **then do**
                    add q' to *new*
                    $\phi \leftarrow$ UNIFY(q', α)
                    if ϕ is not *fail* **then return** ϕ
        add *new* to KB
    return *false*

Properties of forward chaining

- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed
- This is unavoidable: entailment with definite clauses is semi-decidable

Efficiency of forward chaining

- Incremental forward chaining: no need to match a rule on iteration k if a premise wasn't added on iteration k-1
    - match each rule whose premise contains a newly added positive literal
- Matching itself can be expensive:
    - Database indexing allows O(1) retrieval of known facts
    - e.g., query Missile(x) retrieves Missile(M1)
- Forward chaining is widely used in deductive databases

Backward chaining algorithm

function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\ \}$
  local variables: ans, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each r **in** KB where $\textsc{Standardize-Apart}(r) = (\, p_1 \ \wedge \ \ldots \ \wedge \ p_n \ \Rightarrow \ q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $ans \leftarrow \text{FOL-BC-}\textsc{Ask}(KB, [p_1, \ldots, p_n | \textsc{Rest}(goals)], \textsc{Compose}(\theta, \theta')) \cup ans$
  return ans

SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
    - fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
    - fix by using caching of previous results (extra space)
- Widely used for logic programming

Resolution

- Full first-order version:
  $p_1 \vee \cdots \vee p_k, \;\; q_1 \vee \cdots \vee q_n$
  $\models (p_1 \vee \cdots \vee p_{i-1} \vee p_{i+1} \vee \cdots \vee p_k \vee q_1 \vee \cdots \vee q_{j-1} \vee q_{j+1} \vee \cdots \vee q_n)\theta$
  where Unify(p_i, $\neg q_j$) = θ.

- The two clauses are assumed to be standardized apart so that they share no variables.

- For example,
  $\neg$Healthy(x) $\vee$ Happy(x), Healthy(John) $\models$ Happy(John)
  with θ = {x/John}

- Inference: Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$) to see whether it is unsatisfiable.

- Complete for FOL

Conversion to CNF (I)

- Everyone who loves all animals is loved by someone:
  $\forall x\ (\forall y\ Animal(y) \Rightarrow Loves(x, y)) \Rightarrow (\exists y\ Loves(y, x))$

- Eliminate bi-conditionals and implications
  $\forall x\ (\neg \forall y\ \neg Animal(y) \vee Loves(x, y)) \vee (\exists y\ Loves(y, x))$

- Move $\neg$ inwards: $\neg \forall x\ p \equiv \exists x\ \neg p,\ \ \neg\ \exists x\ p \equiv \forall x\ \neg p$
  $\forall x\ (\exists y\ \neg(\neg Animal(y) \vee Loves(x, y))) \vee (\exists y\ Loves(y, x))$
  $\forall x\ (\exists y\ \neg \neg Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y\ Loves(y, x))$
  $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y\ Loves(y, x))$

Conversion to CNF (II)

- Standardize variables: each quantifier should use a different one
  $\forall x\ (\exists y\ Animal(y) \land \neg Loves(x, y)) \lor (\exists z\ Loves(z, x))$
- Skolemize: a more general form of existential instantiation.
  Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
  $\forall x\ (Animal(F(x)) \land \neg Loves(x,F(x))) \lor Loves(G(x),x)$
- Drop universal quantifiers:
  $(Animal(F(x)) \land \neg Loves(x,F(x))) \lor Loves(G(x),x)$
- Distribute $\lor$ over $\land$:
  $(Animal(F(x)) \lor Loves(G(x),x)) \land (\neg Loves(x,F(x)) \lor Loves(G(x),x))$