CS 2740 Knowledge Representation
Lecture 9

First-order logic.
Inference.

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Logical inference in FOL

Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence α , does the KB semantically entail α ?

$$KB \models \alpha \quad ?$$

- In other words: In all interpretations in which sentences in the KB are true, is also α true?

Logical inference problem in the 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.)

Inference in FOL: Truth table approach

- Is the Truth-table approach a viable approach for the FOL?
  ?
- NO!
- Why?
- It would require us to enumerate and list all possible interpretations I
- I = (assignments of symbols to objects, predicates to relations and functions to relational mappings)
- Simply there are too many interpretations

Inference in FOL: Inference rules

- Is the Inference rule approach a viable approach for the FOL?
  ?
- **Yes.**
- The inference rules represent sound inference patterns one can apply to sentences in the KB
- What is derived follows from the KB
- **Caveat:**
  - we need to add rules for handling quantifiers

Inference rules

- **Inference rules from the propositional logic:**
  - Modus ponens

$$\frac{A \Rightarrow B, \quad A}{B}$$

  - Resolution

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

  - and others: And-introduction, And-elimination, Or-introduction, Negation elimination
- **Additional inference rules** are needed for sentences with quantifiers and variables
  - Must involve variable substitutions

Variable substitutions

- Variables in the sentences can be substituted with terms.
  (terms = constants, variables, functions)
- **Substitution:**
  - Is a mapping from **variables to terms**

$$\{x_1 / t_1, x_2 / t_2, \ldots\}$$

  - Application of the substitution to sentences

$$SUBST(\{x / Sam, y / Pam\}, Likes(x, y)) = Likes(Sam, Pam)$$

$$SUBST(\{x / z, y / fatherof(John)\}, Likes(x, y)) =$$
$$Likes(z, fatherof(John))$$

Inference rules for quantifiers

- **Universal elimination**

$$\frac{\forall x\ \phi(x)}{\phi(a)} \qquad a \text{ - is a constant symbol}$$

  – substitutes a variable with **a constant symbol**
- **Example:**

$$\forall x\ Likes(x, IceCream)$$

$$\downarrow$$

$$Likes(Ben, IceCream)$$

Inference rules for quantifiers

- **Existential elimination**

$$\frac{\exists x\ \phi(x)}{\phi(a)}$$

  – Substitutes a variable with **a constant symbol** that does not appear elsewhere in the KB
- **Examples:**
- $\exists x\ Kill(x, Victim) \longrightarrow Kill(Murderer, Victim)$

  Special constant called **Skolem** constant
- $\exists x\ Crown(x) \wedge OnHead(x, John)$

  $Crown(C_1) \wedge OnHead(C_1, John)$

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in **all possible** ways, we have:

  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
  King(John)
  Greedy(John)
  Brother(Richard,John)

- The new KB is **propositionalized**: proposition symbols are

  King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction contd.

- Every FOL KB can be **propositionalized** so as to preserve entailment
- A ground sentence is entailed by new KB iff entailed by the original KB
- **Idea of the inference:**
  – propositionalize **KB and query**,
  – apply resolution, return result
- **Problem:** with function symbols, there are infinitely many ground terms,
  – e.g., *Father*(*Father*(*Father*(*John*)))

Reduction contd.

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

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: Turing (1936), Church (1936) Entailment for FOL is **semidecidable** (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

Problems with propositionalization

- **Propositionalization** seems to generate lots of irrelevant sentences
- **E.g.,** from:
  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  King(John)
  $\forall y\ Greedy(y)$
  Brother(Richard,John)
- It seems obvious that *Evil*(*John*) holds if we want to prove it, but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations.

Unification

- **Problem in inference:** Universal elimination gives many opportunities for substituting variables with ground terms

$$\frac{\forall x\ \phi(x)}{\phi(a)} \qquad a \text{ - is a constant symbol}$$

- **Solution:** Try substitutions that help us to make progress
  - Use substitutions of "similar" sentences in KB
- **Example:**

  $\forall$x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

  King(John)

  $\forall$y Greedy(y)

  If we use a substitution σ = {x/John,y/John}

  we can use modus ponens to infer Evil(John) in one step

Unification.

- **Unification:** takes two similar sentences and computes the substitution that makes them look the same, if it exists

  $UNIFY(p,q) = \sigma$ s.t. $SUBST(\sigma, p) = SUBST(\sigma, q)$
- **Examples:**

$$UNIFY(Knows(John, x), Knows(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = ?$$

$$UNIFY(Knows(John, x), Knows(y, MotherOf(y))) \\ = ?$$

$$UNIFY(Knows(John, x), Knows(x, Elizabeth)) = ?$$

Unification. Examples.

- **Unification:**

  $UNIFY\,(p,q) = \sigma$ s.t. $SUBST(\sigma, p) = SUBST\,(\sigma, q)$

- **Examples:**

$$UNIFY\,(Knows\,(John, x), Knows\,(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x / Ann, y / John\}$$

$$UNIFY\,(Knows\,(John, x), Knows\,(y, MotherOf\,(y)))$$
$$= ?$$

$$UNIFY\,(Knows\,(John, x), Knows\,(x, Elizabeth\,)) = ?$$

Unification. Examples.

- **Unification:**

  $UNIFY\,(p,q) = \sigma$ s.t. $SUBST(\sigma, p) = SUBST\,(\sigma, q)$

- **Examples:**

$$UNIFY\,(Knows\,(John, x), Knows\,(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x / Ann, y / John\}$$

$$UNIFY\,(Knows\,(John, x), Knows\,(y, MotherOf\,(y)))$$
$$= \{x / MotherOf\,(John), y / John\}$$

$$UNIFY\,(Knows\,(John, x), Knows\,(x, Elizabeth\,)) = ?$$

Unification. Examples.

- **Unification:**

  $UNIFY(p,q) = \sigma$ s.t. $SUBST(\sigma, p) = SUBST(\sigma, q)$

- **Examples:**

$$UNIFY(Knows(John, x), Knows(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x / Ann, y / John\}$$

$$UNIFY(Knows(John, x), Knows(y, MotherOf(y))) = \{x / MotherOf(John), y / John\}$$

$$UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail$$

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,
  σ = {y/John, x/z } or σ = {y/John, x/John, z/John}
- The first unifier is **more general** than the second.
- There is a single **most general unifier** (MGU) that is unique up to renaming of variables.
  MGU = { y/John, x/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 inference rules.

- **Use substitutions that let us make inferences**

Example: Modus Ponens

- **If there exists a substitution** σ **such that**

$$SUBST(\sigma, A_i) = SUBST(\sigma, A_i') \quad \textbf{for all i=1,2, n}$$

$$\frac{A_1 \wedge A_2 \wedge \ldots A_n \Rightarrow B, \quad A_1', A_2', \ldots A_n'}{SUBST(\sigma, B)}$$

- Substitution that satisfies the generalized inference rule can be build via unification process
- Advantage of the generalized rules: they are focused
  - only substitutions that allow the inferences to proceed

Generalized Modus Ponens (GMP)

$$\frac{A_1 \wedge A_2 \wedge \ldots A_n \Rightarrow B, \quad A_1', A_2', \ldots A_n'}{SUBST(\sigma, B)}$$

A_1' is *King*(*John*) A_1 is *King*(*x*)
A_2' is *Greedy*(*y*) A_2 is *Greedy*(*x*)
σ is {x/John,y/John} B is *Evil*(*x*)
SUBS(σ,B) = *Evil*(*John*)

- GMP is used with KB of **definite clauses**
- Definite clauses **exactly one positive literal**

- All variables assumed universally quantified

Resolution inference rule

- **Recall:** Resolution inference rule is sound and complete (refutation-complete) for the **propositional logic** and CNF

$$\frac{A \vee B, \quad \neg A \vee C}{B \vee C}$$

- **Generalized resolution rule is sound and refutation complete** for the first-order logic and CNF w/o equalities (if unsatisfiable the resolution will find the contradiction)

$$\sigma = UNIFY\ (\phi_i, \neg \psi_j) \neq fail$$

$$\frac{\phi_1 \vee \phi_2 \ldots \vee \phi_k, \quad \psi_1 \vee \psi_2 \vee \ldots \psi_n}{SUBST(\sigma, \phi_1 \vee \ldots \vee \phi_{i-1} \vee \phi_{i+1} \ldots \vee \phi_k \vee \psi_1 \vee \ldots \vee \psi_{j-1} \vee \psi_{j+1} \ldots \psi_n)}$$

Example: $$\frac{P(x) \vee Q(x), \quad \neg Q(John) \vee S(y)}{P(John) \vee S(y)}$$

Inference with resolution rule

- **Proof by refutation:**
  - Prove that $KB, \neg \alpha$ is **unsatisfiable**
  - resolution is **refutation-complete**

- **Main procedure (steps):**
  1. Convert $KB, \neg \alpha$ to CNF with ground terms and universal variables only
  2. Apply repeatedly the resolution rule while keeping track and consistency of substitutions
  3. Stop when empty set (contradiction) is derived or no more new resolvents (conclusions) follow

Conversion to CNF

1. Eliminate implications, equivalences

$$(p \Rightarrow q) \rightarrow (\neg p \vee q)$$

2. Move negations inside (DeMorgan's Laws, double negation)

$$\neg(p \wedge q) \rightarrow \neg p \vee \neg q$$
$$\neg(p \vee q) \rightarrow \neg p \wedge \neg q$$
$$\neg \forall x \; p \rightarrow \exists x \; \neg p$$
$$\neg \exists x \; p \rightarrow \forall x \; \neg p$$
$$\neg\neg \; p \rightarrow p$$

3. Standardize variables (rename duplicate variables)

$$(\forall x \; P(x)) \vee (\exists x \; Q(x)) \rightarrow (\forall x \; P(x)) \vee (\exists y \; Q(y))$$

Conversion to CNF

4. Move all quantifiers left (no invalid capture possible)

$$(\forall x \; P(x)) \vee (\exists y \; Q(y)) \rightarrow \forall x \; \exists y \; P(x) \vee Q(y)$$

5. Skolemization (removal of existential quantifiers through elimination)

- If no universal quantifier occurs before the existential quantifier, replace the variable with a new constant symbol

$$\exists y \; P(A) \vee Q(y) \rightarrow P(A) \vee Q(B)$$

- If a universal quantifier precede the existential quantifier replace the variable with a function of the "universal" variable

$$\forall x \; \exists y \; P(x) \vee Q(y) \rightarrow \forall x \;\; P(x) \vee Q(F(x))$$

$F(x)$ - **a Skolem function**

Conversion to CNF

6. Drop universal quantifiers (all variables are universally quantified)

$$\forall x \;\; P(x) \vee Q(F(x)) \rightarrow P(x) \vee Q(F(x))$$

7. Convert to CNF using the distributive laws

$$p \vee (q \wedge r) \rightarrow (p \vee q) \wedge (p \vee r)$$

The result is a CNF with variables, constants, functions

Resolution example

$$\overbrace{\neg P(w) \vee Q(w), \neg Q(y) \vee S(y),\; P(x) \vee R(x), \neg R(z) \vee S(z)}^{\textbf{KB}},\;\; \overbrace{\neg S(A)}^{\neg \alpha}$$

Resolution example

KB $\qquad\qquad$ $\neg\alpha$

$$\neg P(w) \vee Q(w), \neg Q(y) \vee S(y), \; P(x) \vee R(x), \neg R(z) \vee S(z), \quad \neg S(A)$$

$\{y / w\}$

$$\neg P(w) \vee S(w)$$

Resolution example

KB $\qquad\qquad$ $\neg\alpha$

$$\neg P(w) \vee Q(w), \neg Q(y) \vee S(y), \; P(x) \vee R(x), \neg R(z) \vee S(z), \quad \neg S(A)$$

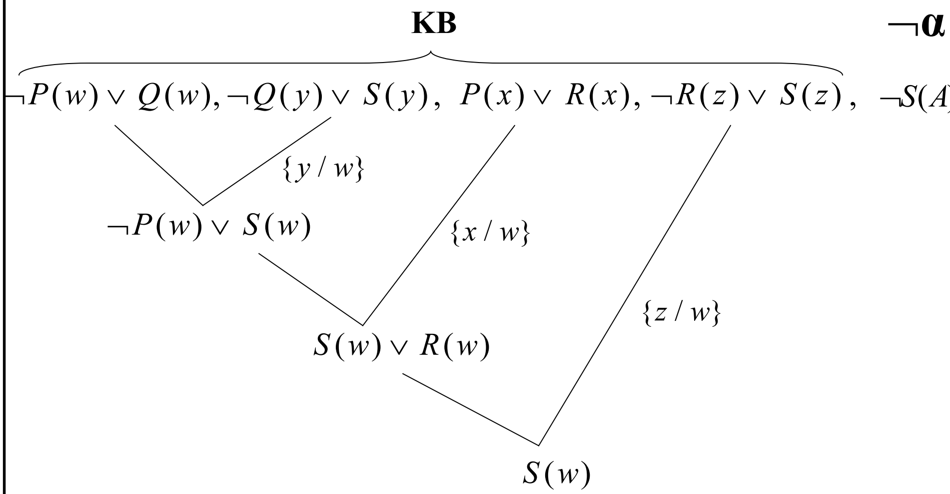
$\{y / w\}$

$$\neg P(w) \vee S(w)$$

$\{x / w\}$

$$S(w) \vee R(w)$$

Resolution example



Resolution example