Artificial Intelligence

CS4365 --- Fall 2022

Predicate Logic

Instructor: Yunhui Guo

KR Language: Predicate Logic

- Gives us a more concise formulation.
- Essentially equivalent to propositional logic in finite domains.
- Extends propositional logic with variables, predicates, symbols, functions, and quantifiers ($\forall$, $\exists$).
- Key properties from propositional case carry over: model-theoretic semantics, sound and complete proof theory, sound and refutation complete resolution.

Inference in Predicate Logic

- How do we reason with predicate (first-order) logic? Derive new info?

- We can use resolution as in propositional case:

  From $(\alpha \vee p) \wedge (\neg p \vee \beta)$

  conclude $\alpha \vee \beta$ until you reach contradiction

- First-order inference can be done by converting the knowledge base to propositional logic

- But we need some extra "tricks" to deal with quantifiers and variables

Inference Rules for Quantifiers

- $\forall$ 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)
  - ...
- **Universal Instantiation (UI)**: we can infer any sentence obtained by substituting a **ground term g** (a term without variables) for the variable v.

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

{x/John}, {x/Richard}, ...

Inference Rules for Quantifiers

- $\exists$ x Crown(x) $\wedge$ OnHead(x, John)

- **Existential Instantiation**: the variable is replaced by a single new constant symbol k

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

- k that does not appear elsewhere in the knowledge base (a **Skolem** constant)
- Crown(C1) $\wedge$ OnHead(C1, John)

Inference rules for quantifiers

- Universal Instantiation can be applied many times to produce many propositions
- Existential Instantiation can be applied once, and then the existentially quantified sentence can be discarded
  - Once we add $Crown(C1) \land OnHead(C1, John)$, we don't need $\exists x\ Crown(x) \land OnHead(x, John)$

Reduction to Propositional Inference

- Knowledge base:
  - $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
  - King(John)
  - Greedy(John)
  - Brother (Richard, John)
- A universally quantified sentence can be replaced by the set of all possible instantiations
  - King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  - King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard) ,

Propositionalization

Reduction to Propositional Inference

- The set of possible ground-term substitutions is infinite if there is a function symbol
  - Father (Father (Father (John)))
- **Theorem**: If a sentence is entailed by the original, first-order knowledge base, then there is a proof involving just a finite subset of the propositionalized knowledge base.
- First-order inference via propositionalization that is complete
- What happens when the sentence is not entailed?
  - Cannot determine if a sentence is not entailed

Reduction to Propositional Inference

- Can be inefficient

- $\forall$ 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)
    - ...

- To infer Evil(John) there is need to generate King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

Lifting

- $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
- Find substitution θ that makes each of the conjuncts of the premise of the implication identical to sentences already in the knowledge base, then we can assert the conclusion of the implication
- {x/John} achieves the aim

Lifting

- $\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
- If we know $\forall$y Greedy(y)
- We need to find a substitution both for the variables in the implication sentence and for the variables in the sentences that are in the knowledge base
- {x/John, y/John} achieves the aim

Lifting

- **Generalized Modus Ponens**
  - For atomic sentences p_i, $p_{i'}$ and q, where there is a substitution θ such that SUBST(θ, $p_{i'}$) = SUBST(θ, p_i) for all i,

$$\frac{p'_1, p'_2, \ldots, p'_n, \ (p_1 \wedge p_2 \wedge \ldots, \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

There n+1 premises: the n atomic sentences $p_{i'}$ and one implication

Lifting

- In our example:
  - $p_{1'}$ is King(John) p_1 is King(x)
  - $p_{2'}$ is Greedy(y) p_2 is Greedy(x)
  - θ is {x/John, y/John}   q is Evil(x)

  - SUBST(θ ,q) is Evil(John)

- Generalized Modus Ponens is sound
- Generalized Modus Ponens is a lifted version of Modus Ponens

Unification

UNIFY(P,Q) takes two atomic sentences P and Q and returns a substitution that makes P and Q look the same.

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

Unifier: a substitution that makes two clauses resolvable.

v1/C; v2/v3; v4/f(...)

Unification -- Purpose

Given:

Know(John, x) → Hates(John, x)

Knows(John, Jim)

Derive:

Hates(John, Jim)

Need **unifier** {x/Jim} for resoluion to work.

(simplest case)

Unification (example)

one rule:

Know(John, x) → Hates(John, x)

facts:

Knows(John, Jim)

Knows(y, leo)

Knows(z, Mother(z))

Knows(x, Jane)

Who does John hate?

Most General Unifier

In cases where there is more than one substitution choose the one that makes the least commitment (most general) about the bindings.

UNIFY(Knows(John, x), Knows(y, z))

= {y/John, x/z}

or {y/John, x/z, z/Freda}

or {y/John, x/John, z/John}

or ...

See R&N for general unification algorithm

Unification

- Finding substitutions that make different logical expressions look identical
- UNIFY(Knows(John, x), Knows(x, Elizabeth))
  - fail

- The problem can be avoided by standardizing apart one of the two sentences being unified, which means renaming its variables to avoid name clashes.
- UNIFY(Knows(John, x), Knows(x_{17}, Elizabeth)) = {x/Elizabeth, x_{17}/John}

Resolution

I put KB in CNF form

all variables universally quantified

main trick: “skolemization” to remove existentials

idea: invent names for unknown objects known to exist

II use unification to match atomic sentences

III apply resolution rule to the CNF combined
with negated goal. Attempt to generate empty clause

Converting more complicated axioms to CNF

Axiom:

$\forall x$: brick(x) $\rightarrow$ (($\exists y$: on(x, y) $\wedge$ $\neg$pyramid(y))
$\wedge$ ($\neg\exists y$: on(x, y) $\wedge$ on(y, x))
$\wedge$ ($\forall y$: $\neg$brick(y) $\rightarrow$ $\neg$equal(x, y)))

$\neg$brick(x) $\vee$ on(x, support(x))

$\neg$brick(w) $\vee$ $\neg$pyramid(support(w))

$\neg$brick(u) $\vee$ $\neg$on(u, y) $\vee$ $\neg$on(y ,u)

$\neg$brick(v) $\vee$ brick(z) $\vee$ $\neg$equal(v, z)

1. Eliminate implications

Substitute $\neg E_1 \vee E_2$ for $E_1 \rightarrow E_2$

$$
\begin{aligned}
\forall x: \text{brick}(x) \rightarrow (&(\exists y: \text{on}(x, y) \wedge \neg\text{pyramid}(y)) \\
&\wedge (\neg\exists y: \text{on}(x, y) \wedge \text{on}(y, x)) \\
&\wedge (\forall y: \neg\text{brick}(y) \rightarrow \neg\text{equal}(x, y)))
\end{aligned}
$$

$$
\begin{aligned}
\forall x: \neg\text{brick}(x) \vee (&(\exists y: \text{on}(x, y) \wedge \neg\text{pyramid}(y)) \\
&\wedge (\neg\exists y: \text{on}(x, y) \wedge \text{on}(y, x)) \\
&\wedge (\forall y: \neg(\neg\text{brick}(y)) \vee \neg\text{equal}(x, y)))
\end{aligned}
$$

2. Move negations down to the atomic formulas

$\neg(E_1 \wedge E_2) \Leftrightarrow (\neg E_1) \vee (\neg E_2)$

$\neg(E_1 \vee E_2) \Leftrightarrow (\neg E_1) \wedge (\neg E_2)$

$\neg(\neg E_1) \Leftrightarrow E_1$

$\neg \forall x: E_1(x) \Leftrightarrow \exists x: \neg E_1(x)$

$\neg \forall x: E_1(x) \Leftrightarrow \forall x: \neg E_1(x)$

$\forall x: \neg brick(x) \vee$

$((\exists y: on(x, y) \wedge \neg pyramid(y))$

$\wedge (\neg \exists y: on(x, y) \wedge on(y, x))$

$\wedge (\forall y: \neg(\neg brick(y)) \vee \neg equal(x, y)))$

3. Eliminate existential quantifiers

Skolemization

Harder cases:

$\forall x$: $\exists y$: father(y, x) become $\forall x$: father(S1(x), x)

There is one argument for each universally quantified variable whose scope contains the Skolem function.

Easy cases:

$\exists x$: President(x) becomes President(S2)

$\forall x$: $\neg$brick(x) $\vee$ (($\exists y$: on(x, y) $\wedge$ $\neg$pyramid(y)) $\wedge$ ...

4. Rename variables as necessary

We want no two variables of the same name.

$$
\begin{aligned}
\forall x: \neg brick(x) \vee ((& on(x, S1(x)) \wedge \neg pyramid(S1(x))) \\
& \wedge (\forall y: (\neg on(x, y) \vee \neg on(y, x))) \\
& \wedge (\forall y: (brick(y) \vee \neg equal(x, y))))
\end{aligned}
$$

$$
\begin{aligned}
\forall x: \neg brick(x) \vee ((& on(x, S1(x)) \wedge \neg pyramid(S1(x))) \\
& \wedge (\forall y: (\neg on(x, y) \vee \neg on(y, x))) \\
& \wedge (\forall z: (brick(z) \vee \neg equal(x, z))))
\end{aligned}
$$

5. Move the universal quantifiers to the left

This works because each quantifier uses a unique variable name.

$$\forall x: \neg brick(x) \vee ((on(x, S1(x)) \wedge \neg pyramid(S1(x)))$$
$$\wedge (\forall y: (\neg on(x, y) \vee \neg on(y, x)))$$
$$\wedge (\forall z: (brick(z) \vee \neg equal(x, z))))$$

$$\forall x \forall y \forall z: \neg brick(x) \vee ((on(x, S1(x)) \wedge \neg pyramid(S1(x)))$$
$$\wedge (\neg on(x, y) \vee \neg on(y, x))$$
$$\wedge (brick(z) \vee \neg equal(x, z)))$$

6.Move disjunctions down to the literals

$E_1 \vee (E_2 \wedge E_3) \Leftrightarrow (E_1 \vee E_2) \wedge (E_1 \wedge E_3)$

$$\begin{aligned}\forall x \forall y \forall z: & (\neg brick(x) \vee (on(x, S1(x)) \wedge \neg pyramid(S1(x))))\\ & \wedge (\neg brick(x) \vee \neg on(x, y) \vee \neg on(y, x))\\ & \wedge (\neg brick(x) \vee brick(z) \vee \neg equal(x, z))\end{aligned}$$

$$\begin{aligned}\forall x \forall y \forall z: & (\neg brick(x) \vee (on(x, S1(x)))\\ & \wedge (\neg brick(x) \vee \neg pyramid(S1(x)))\\ & \wedge (\neg brick(x) \vee \neg on(x, y) \vee \neg on(y, x))\\ & \wedge (\neg brick(x) \vee brick(z) \vee \neg equal(x, z))\end{aligned}$$

7.Eliminate the conjunctions

$\forall x \forall y \forall z$: $(\neg brick(x) \lor (on(x, S1(x)))$
$\land (\neg brick(x) \lor \neg pyramid(S1(x)))$
$\land (\neg brick(x) \lor \neg on(x, y) \lor \neg on(y, x))$
$\land (\neg brick(x) \lor brick(z) \lor \neg equal(x, z))$

$\forall x$: $\neg brick(x) \lor (on(x, S1(x))$
$\forall x$: $\neg brick(x) \lor \neg pyramid(S1(x))$
$\forall x \forall y$: $\neg brick(x) \lor \neg on(x, y) \lor \neg on(y, x)$
$\forall x \forall z$: $\neg brick(x) \lor brick(z) \lor \neg equal(x, z)$

8.Rename all variables, as necessary, so no two have the same name

$\forall x: \neg brick(x) \lor (on(x, S1(x))$

$\forall x: \neg brick(x) \lor \neg pyramid(S1(x))$

$\forall x \forall y: \neg brick(x) \lor \neg on(x, y) \lor \neg on(y, x)$

$\forall x \forall z: \neg brick(x) \lor brick(z) \lor \neg equal(x, z)$

$\forall x: \neg brick(x) \lor (on(x, S1(x))$

$\forall w: \neg brick(w) \lor \neg pyramid(S1(w))$

$\forall u \forall y: \neg brick(u) \lor \neg on(u, y) \lor \neg on(y, u)$

$\forall v \forall z: \neg brick(v) \lor brick(z) \lor \neg equal(v, z)$

9.Eliminate the universal quantifiers

At this point, all remaining variables must be universally quantified.

$\neg brick(x) \lor (on(x, S1(x))$

$\neg brick(w) \lor \neg pyramid(S1(w))$

$\neg brick(u) \lor \neg on(u, y) \lor \neg on(y, u)$

$\neg brick(v) \lor brick(z) \lor \neg equal(v, z)$

Algorithm: Putting Axioms into CNF

- Eliminate the implications.
- Move the negations down to the atomic formulas.
- Eliminate the existential quantifiers.
- Rename the variables, if necessary.
- Move the universal quantifiers to the left.
- Move the disjunctions down to the literals.
- Eliminate the conjunctions.
- Rename the variables, if necessary.
- Eliminate the universal quantifiers.

Example

- Everyone who loves all animals is loved by someone

- $\forall x\ [\forall y\text{: } Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$

Eliminate the implications

- $\forall x\ [\forall y{:}\ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \neg\forall y{:}\ \neg Animal(y)\ \vee\ Loves(x, y)]\ \vee\ [\exists y\ Loves(y, x)]$

Move ¬ Inwards

- $\forall x\ [\forall y: Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$
- $\forall x\ [\neg\forall y: \neg Animal(y) \lor Loves(x, y)] \lor [\exists y\ Loves(y, x)]$

$\neg\forall x\ P(x) \Leftrightarrow \exists x\ \neg P(x)$

$\neg\exists x\ P(x) \Leftrightarrow \forall x\ \neg P(x)$

i. $\forall x\ [\exists y: \neg(\neg Animal(y) \lor Loves(x, y))] \lor [\exists y\ Loves(y, x)]$

ii. $\forall x\ [\exists y: \neg\neg Animal(y) \land \neg Loves(x, y)] \lor [\exists y\ Loves(y, x)]$

iii. $\forall x\ [\exists y: Animal(y) \land \neg Loves(x, y)] \lor [\exists y\ Loves(y, x)]$

Standardize Variables

- $\forall x\ [\forall y: Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \neg\forall y: \neg Animal(y)\ \lor\ Loves(x, y)]\ \lor\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y: Animal(y)\ \land\ \neg Loves(x, y)]\ \lor\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y: Animal(y)\ \land\ \neg Loves(x, y)]\ \lor\ [\exists z\ Loves(z, x)]$

Skolemize

- $\forall x\ [\forall y\text{: Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y\ \text{Loves}(y, x)]$
- $\forall x\ [\ \neg\forall y\text{: }\neg\text{Animal}(y)\ \lor\ \text{Loves}(x, y)]\ \lor\ [\exists y\ \text{Loves}(y, x)]$
- $\forall x\ [\ \exists y\text{: Animal}(y)\ \land\ \neg\text{Loves}(x, y)]\ \lor\ [\exists y\ \text{Loves}(y, x)]$
- $\forall x\ [\ \exists y\text{: Animal}(y)\ \land\ \neg\text{Loves}(x, y)]\ \lor\ [\exists z\ \text{Loves}(z, x)]$

- $\forall x\ [\ \text{Animal}(F(x))\ \land\ \neg\text{Loves}(x, F(x))]\ \lor\ [\text{Loves}(G(z), x)]$

- Skolem functions: F and G

Drop universal quantifiers

- $\forall x\ [\forall y\!: Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \neg\forall y\!: \neg Animal(y)\ \vee\ Loves(x, y)]\ \vee\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y\!: Animal(y)\ \wedge\ \neg Loves(x, y)]\ \vee\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y\!: Animal(y)\ \wedge\ \neg Loves(x, y)]\ \vee\ [\exists z\ Loves(z, x)]$
- $\forall x\ [\ Animal(F(x))\ \wedge\ \neg Loves(x, F(x))]\ \vee\ [Loves(G(z), x)]$
- $[Animal(F(x))\ \wedge\ \neg Loves(x, F(x))]\ \vee\ [Loves(G(z), x)]$

Distribute $\vee$ over $\wedge$

- $\forall x\ [\forall y\text{: } Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \neg\forall y\text{: } \neg Animal(y)\ \vee\ Loves(x, y)]\ \vee\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y\text{: } Animal(y)\ \wedge\ \neg Loves(x, y)]\ \vee\ [\exists y\ Loves(y, x)]$
- $\forall x\ [\ \exists y\text{: } Animal(y)\ \wedge\ \neg Loves(x, y)]\ \vee\ [\exists z\ Loves(z, x)]$
- $\forall x\ [\ Animal(F(x))\ \wedge\ \neg Loves(x, F(x))]\ \vee\ [Loves(G(z), x)]$
- $[Animal(F(x))\ \wedge\ \neg Loves(x, F(x))]\ \vee\ [Loves(G(z), x)]$
- $[Animal(F(x))\ \vee\ Loves(G(z), x)]\ \wedge [\ \neg Loves(x, F(x))\ \vee\ [Loves(G(z), x)]$

- Now the KB is in CNF

The resolution inference rule

- Two clauses, which are assumed to be standardized apart so that they share no variables, can be resolved if they contain complementary literals
- First-order literals are complementary if one unifies with the negation of the other.
- From $l_1 \vee l_2, \ldots, \vee l_k$ and $m_1 \vee m_2 \ldots \vee m_n$, if $\text{UNIFY}(l_i, \neg m_j) = \theta$, we have

$$\text{SUBST}(\theta, l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee m_2 \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)$$

The resolution inference rule

- Example:
  - [Animal(F(x)) $\lor$ Loves(G(x), x)]
  - [¬Loves(u, v) $\lor$ ¬Kills(u, v)]

  - Resolve by eliminating the complementary literals Loves(G(x), x) and ¬Loves(u, v), with unifier $\theta = \{u/G(x), v/x\}$,

  - We have the resolvent [Animal(F(x)) $\lor$ ¬Kills(G(x), x)] .

Completeness

- Resolution with unification applied to CNF a complete inference procedure.
- In practice, still a significant search problem!
- Many different search strategies: **resolution strategies**.

Strategies for Selecting Clauses

- Unit-preference strategy: Give preference to resolutions involving the clauses with the smallest number of literals.
  - More likely to produce empty clause
- Set-of-support strategy: Try to resolve with the negated theorem or a clause generated by resolution from that clause.
- **Subsumption**: Eliminates all sentences that are subsumed (i.e., more specific than) an existing sentence in the KB
  - If P(x) is in the KB, then there is no need to have P(A)

Logic programming

- Expressing knowledge in a formal language and that problems is solved by running inference
- Algorithm = Logic + Control
- Prolog:
  - The most widely used logic programming language
  - Different notations from predicate logic
  - $A \land B \Rightarrow C \quad \rightarrow \quad C :- A, B$
  - The execution of Prolog programs is based on inference