Using Predicate Logic

Chapter 5

Using Propositional Logic

Representing simple facts

It is raining
RAINING

It is sunny
SUNNY

It is windy
WINDY

If it is raining, then it is not sunny
RAINING $\rightarrow \neg$SUNNY

Using Propositional Logic

- Theorem proving is decidable
- Cannot represent objects and quantification

Using Predicate Logic

- Can represent objects and quantification
- Theorem proving is semi-decidable

Predicate Logic Syntax

- Constant symbols: a, b, c, John, …
  to represent primitive objects
- Variable symbols: x, y, z, …
  to represent unknown objects
- Predicate symbols: safe, married, love, …
  to represent relations

married(John)

love(John, Mary)

Predicate Logic Syntax

- Function symbols: square, father, ...

  to represent simple objects

  safe(square(1, 2))

  love(father(John), mother(John))

- Terms:

  to represent complex objects

  – Constant symbols

  – If f is a function symbol, and t_1, t_2, ..., t_n are terms, then so is $f(t_1, t_2, ..., t_n)$

  love(mother(father(John)), John)

Predicate Logic Syntax

- Logical connectives: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Universal quantifier: $\forall x$: p(x)

  $\forall x$: love(father(x), mother(x))

- Existential quantifier: $\exists x$: p(x) $\equiv \neg\forall x$: $\neg$p(x)

  $\exists x$: $\neg$married(x)

Predicate Logic Syntax

- Sentences:
  - Atomic sentences: $p(t_1, t_2, \ldots, t_n)$
  - If α is a sentence, then so are $\neg\alpha$ and (α)
  - If α and β are sentences, then so are $\alpha \land \beta$, $\alpha \lor \beta$, $\alpha \Rightarrow \beta$, and $\alpha \Leftrightarrow \beta$
  - If α is a sentence, then so are $\forall\alpha$ and $\exists\alpha$

Using Predicate Logic

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Pompeians were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

Using Predicate Logic

1. Marcus was a man.

   man(Marcus)

Using Predicate Logic

2. Marcus was a Pompeian.

   Pompeian(Marcus)

Using Predicate Logic

3. All Pompeians were Romans.

   $\forall x$: Pompeian(x) $\rightarrow$ Roman(x)

Using Predicate Logic

4. Caesar was a ruler.

   ruler(Caesar)

Using Predicate Logic

5. All Pompeians were either loyal to Caesar or hated him.

   inclusive-or

   $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar)

   exclusive-or

   $\forall x$: Roman(x) $\rightarrow$ (loyalto(x, Caesar) $\wedge$ $\neg$hate(x, Caesar)) $\vee$

   ($\neg$loyalto(x, Caesar) $\wedge$ hate(x, Caesar))

Using Predicate Logic

6. Every one is loyal to someone.

   $\forall x: \exists y: loyalto(x, y)$ $\exists y: \forall x: loyalto(x, y)$

Using Predicate Logic

7. People only try to assassinate rulers they are not loyal to.

$$\forall x{:}\ \forall y{:}\ person(x) \wedge ruler(y) \wedge tryassassinate(x, y) \rightarrow \neg loyalto(x, y)$$

Using Predicate Logic

7. People only try to assassinate rulers they are not loyal to.

$$\forall x: \forall y: person(x) \wedge ruler(y) \wedge tryassassinate(x, y)$$
$$\rightarrow \neg loyalto(x, y)$$

Using Predicate Logic

8. Marcus tried to assassinate Caesar.

   tryassassinate(Marcus, Caesar)

Using Predicate Logic

Was Marcus loyal to Caesar?

man(Marcus)

ruler(Caesar)

tryassassinate(Marcus, Caesar)

$\Downarrow$ $\forall x$: man(x) $\rightarrow$ person(x)

$\neg$loyalto(Marcus, Caesar)

Using Predicate Logic

- Many English sentences are ambiguous.
- There is often a choice of how to represent knowledge.

Reasoning

1. Marcus was a Pompeian.
2. All Pompeians died when the volcano erupted in 79 A.D.
3. It is now 2008 A.D.

Is Marcus alive?

Reasoning

1. Marcus was a Pompeian.

   Pompeian(Marcus)

2. All Pompeians died when the volcano erupted in 79 A.D.

   $\text{erupted(volcano, 79)} \wedge \forall x\text{: Pompeian}(x) \rightarrow \text{died}(x, 79)$

3. It is now 2008 A.D.

   now = 2008

Reasoning

1. Marcus was a Pompeian.

   $Pompeian(Marcus)$

2. All Pompeians died when the volcano erupted in 79 A.D.

   $erupted(volcano, 79) \wedge \forall x: Pompeian(x) \rightarrow died(x, 79)$

3. It is now 2008 A.D.

   $now = 2008$

$$\forall x: \forall t_1: \forall t_2: died(x, t_1) \wedge greater\text{-}than(t_2, t_1) \rightarrow dead(x, t_2)$$

Reasoning

- Obvious information may be necessary for reasoning
- We may not know in advance which statements to deduce (P or $\neg$P).

Reasoning

KB $\models \alpha$ (α is a logical consequence of KB)

How to prove it automatically?

Resolution

Robinson, J.A. 1965. A machine-oriented logic based on the resolution principle. Journal of ACM 12 (1): 23-41.

Resolution

Proof by refutation

$$KB \models \alpha \quad \Leftrightarrow \quad KB \wedge \neg\alpha \models \text{false (empty clause)}$$

Resolution

Resolution inference rule

$$\frac{(\alpha \vee \neg\beta) \wedge (\gamma \vee \beta) \quad \text{premise}}{(\alpha \vee \gamma) \quad \text{conclusion}}$$

Resolution in Propositional Logic

1. Convert all the propositions of KB to clause form (S).

$$L_1 \lor L_2 \lor \ldots \lor L_n$$

P or $\neg$P

Resolution in Propositional Logic

1. Convert all the propositions of KB to clause form (S).
2. Negate α and convert it to clause form. Add it to S.
3. Repeat until either a contradiction is found or no progress can be made:
    a. Select two clauses $(\alpha \vee \neg P)$ and $(\gamma \vee P)$.
    b. Add the resolvent $(\alpha \vee \gamma)$ to S.

Resolution in Propositional Logic

Example:

KB = {P, $(P \land Q) \rightarrow R$, $(S \lor T) \rightarrow Q$, T}

α = R

Resolution in Predicate Logic

Example:

KB = {P(a), $\forall x$: $(P(x) \wedge Q(x)) \rightarrow R(x)$, $\forall y$: $(S(y) \vee T(y)) \rightarrow Q(y)$, T(a)}

α = R(a)

Resolution in Predicate Logic

Unification:

UNIFY(p, q) = unifier θ where $\theta(p) = \theta(q)$

Resolution in Predicate Logic

Unification:

$\forall x$: knows(John, x) $\rightarrow$ hates(John, x)

knows(John, Jane)

$\forall y$: knows(y, Leonid)

$\forall y$: knows(y, mother(y))

$\forall x$: knows(x, Elizabeth)

Resolution in Predicate Logic

Unification:

$\forall x$: knows(John, x) $\rightarrow$ hates(John, x)
knows(John, Jane)
$\forall y$: knows(y, Leonid)
$\forall y$: knows(y, mother(y))
$\forall x$: knows(x, Elizabeth)

UNIFY(knows(John, x), knows(John, Jane)) = {Jane/x}
UNIFY(knows(John, x), knows(y, Leonid)) = {Leonid/x, John/y}
UNIFY(knows(John, x), knows(y, mother(y))) = {John/y, mother(John)/x}
UNIFY(knows(John, x), knows(x, Elizabeth)) = FAIL

Resolution in Predicate Logic

Unification: Standardization

UNIFY(knows(John, x), knows(y, Elizabeth)) = {John/y, Elizabeth/x}

Resolution in Predicate Logic

Unification: Occur check

UNIFY(knows(x, x), knows(y, mother(y))) = FAIL

Resolution in Predicate Logic

Unification: Most general unifier

UNIFY(knows(John, x), knows(y, z)) = {John/y, John/x, John/z}
= {John/y, Jane/x, Jane/z}
= {John/y, v/x, v/z}
= {John/y, z/x, Jane/v}
= {John/y, z/x}

Conversion to Clause Form

1. Eliminate $\rightarrow$.

   $P \rightarrow Q \equiv \neg P \vee Q$

2. Reduce the scope of each $\neg$ to a single term.

   $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
   $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
   $\neg\forall x{:}\ P \equiv \exists x{:}\ \neg P$
   $\neg\exists x{:}\ p \equiv \forall x{:}\ \neg P$
   $\neg\neg\ P \equiv P$

3. Standardize variables so that each quantifier binds a unique variable.

   $(\forall x{:}\ P(x)) \vee (\exists x{:}\ Q(x)) \equiv (\forall x{:}\ P(x)) \vee (\exists y{:}\ Q(y))$

Conversion to Clause Form

4. Move all quantifiers to the left without changing their relative order.

   $\forall x: (P(x) \vee \exists y: Q(y)) \equiv \forall x: \exists y: (P(x) \vee (Q(y))$

   $(\forall x: P(x)) \vee (\exists y: Q(y))$: don't move!

5. Eliminate $\exists$ (Skolemization).

   $\exists x: P(x) \equiv P(c)$ Skolem constant
   $\forall x: \exists y\ P(x, y) \equiv \forall x: P(x, f(x))$ Skolem function

6. Drop $\forall$.

   $\forall x: P(x) \equiv P(x)$

7. Convert the formula into a conjunction of disjuncts.

   $(P \wedge Q) \vee R \equiv (P \vee R) \wedge (Q \vee R)$

8. Create a separate clause corresponding to each conjunct.

9. Standardize apart the variables in the set of obtained clauses.

Conversion to Clause Form

1. Eliminate $\rightarrow$.
2. Reduce the scope of each $\neg$ to a single term.
3. Standardize variables so that each quantifier binds a unique variable.
4. Move all quantifiers to the left without changing their relative order.
5. Eliminate $\exists$ (Skolemization).
6. Drop $\forall$.
7. Convert the formula into a conjunction of disjuncts.
8. Create a separate clause corresponding to each conjunct.
9. Standardize apart the variables in the set of obtained clauses.

Resolution in Predicate Logic

1. Convert all the propositions of KB to clause form (S).
2. Negate α and convert it to clause form. Add it to S.
3. Repeat until a contradiction is found:
   a. Select two clauses $(\alpha \lor \neg p(t_1, t_2, \ldots, t_n))$ and $(\gamma \lor p(t'_1, t'_2, \ldots, t'_n))$.
   b. $\theta = mgu(p(t_1, t_2, \ldots, t_n), p(t'_1, t'_2, \ldots, t'_n))$
   c. Add the resolvent $\theta(\alpha \lor \gamma)$ to S.

Resolution in Predicate Logic

Example:

KB = $\{P(a), \forall x: (P(x) \wedge Q(x)) \rightarrow R(x), \forall y: (S(y) \vee T(y)) \rightarrow Q(y), T(a)\}$

$\alpha = R(a)$

Example

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Pompeians were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

Example

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\exists y$: loyalto(x, y).
7. $\forall x$: $\forall y$: person(x) $\wedge$ ruler(y) $\wedge$ tryassassinate(x, y)
   $\rightarrow$ $\neg$loyalto(x, y).
8. tryassassinate(Marcus, Caesar).

Example

Prove:

hate(Marcus, Caesar)

Question Answering

1. When did Marcus die?
2. Whom did Marcus hate?
3. Who tried to assassinate a ruler?
4. What happen in 79 A.D.?.
5. Did Marcus hate everyone?

Soundness and Completeness

- Soundness of a reasoning algorithm/system R:

  if KB derives α using R, then KB $\models \alpha$

Soundness and Completeness

- Completeness of a reasoning algorithm/system R:

  if KB $\models \alpha$, then KB derives α using R

Soundness and Completeness

Resolution algorithm is sound and complete

Soundness and Completeness

- In general:
  - Soundness: any returned answer is a correct answer.
  - Completeness: all correct answers are returned.

Programming in Logic

PROLOG:

- Only Horn sentences are acceptable

$$A \leftarrow B_1, B_2, \ldots, B_m \equiv A \vee \neg B_1 \vee \neg B_2 \vee \ldots \vee \neg B_m$$

A, B_i: atoms

Programming in Logic

PROLOG:

- The occur-check is omitted from the unification: unsound

  test $\leftarrow$ P(x, x)

  P(x, f(x))

Programming in Logic

PROLOG:

- Backward chaining with depth-first search: incomplete

$P(x, y) \leftarrow Q(x, y)$

$P(x, x)$

$Q(x, y) \leftarrow Q(y, x)$

Programming in Logic

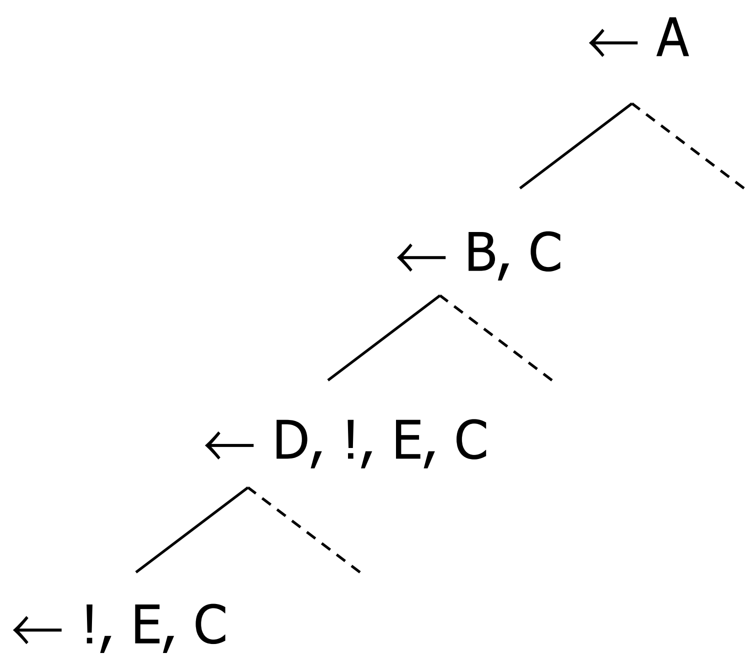
PROLOG:

- Unsafe cut: incomplete

$A \leftarrow B, C$

$B \leftarrow D, !, E$

$D \leftarrow$



Programming in Logic

PROLOG:

- Negation as failure: $\neg$P if fails to prove P

Homework

Exercises

1-13, Chapter 5, Rich&Knight AI Textbook

Chapter 4 of the Vietnamese Textbook