Logic

Knowledge Representation & Reasoning Mechanisms

Logic

- Logic as KR
  - Propositional Logic
  - Predicate Logic (predicate Calculus)

- Automated Reasoning
  - Logical inferences
  - Resolution and Theorem-proving
Logic

● Logic as KR
  ■ Propositional Logic
  ■ Predicate Logic (predicate Calculus)

● Automated Reasoning
  ■ Logical inferences
  ■ Resolution and Theorem-proving

Propositional Logic

● Symbols:
  ■ truth symbols: true, false
  ■ propositions: a statement that is “true” or “false” but not both
    E.g., P = “Two plus two equals four”
    Q = “It rained yesterday.”
  ■ connectives: ~, →, ∧, ∨, ≡

● Sentences - propositions or truth symbols
● Well formed formulas (expressions) - sentences that are legally well-formed with connectives
  E.g., P ∧ R → and P ~ are not wff but P ∧ R → ~ Q is
**Examples**

AI is hard but it is interesting  \( P \land Q \)
AI is neither hard nor interesting \( \neg P \land \neg Q \)

If you don’t do assignments then you will fail \( P \rightarrow Q \)
≡ Do assignments or fail (Prove by truth table) \( \neg P \lor Q \)

None or both of \( P \) and \( Q \) is true \( (\neg P \land \neg Q) \lor (P \land Q) = T \)

Exactly one of \( P \) and \( Q \) is true \( (\neg P \land Q) \lor (P \land \neg Q) = T \)

**Predicate Logic**

- **Symbols:**
  - *truth symbols*
  - *constants:* represents objects in the world
  - *variables:* represents ranging objects
  - *functions:* represent properties
  \( \{ \text{Terms}\} \)

- *Predicates:* functions of terms with true/false values e.g.,
  - `bill_residence_city` (vancouver) or `lives` (bill, vancouver)

- **Atomic sentences:** true, false, or predicates

- **Quantifiers:** \( \forall, \exists \)

- **Sentences (expressions):** sequences of legal applications of connectives and quantifiers to atomic sentences or sentences
Examples

plus(two, three): a function, not a predicate \(\rightarrow\) not an atomic sentence

equal(plus(two, three), five): a predicate \(\rightarrow\) an atomic sentence

lives(bill, vancouver): a predicate \(\rightarrow\) atomic sentence

\(\forall X \ [\text{lives(bill, } X) \rightarrow \text{near ocean}(X)]\): a sentence (expression)

\(\forall Y \ [X(a, Y)]\): a first-order predicate expression

since the quantified variable Y refers to objects and not predicates

\(\forall X \ [X(a, Y)]\): not a first-order predicate expression

More Examples

Block World Problem

\(\text{on}(D, C) \land \text{on}(C, B) \land\)
\(\text{ontable}(B) \land \text{ontable}(A) \land\)
\(\text{clear}(D) \land \text{clear}(A)\)

- Rule describes that a block is clear
  \(\forall X \ [\sim \exists \text{on}(Y, X) \rightarrow \text{clear}(X)]\)

- To stack X on Y:
  \(\forall X \forall Y \ [\text{hand_empty} \land \text{clear}(Y) \land\)
  \(\text{pickup}(X) \land \text{putdown}(X,Y) \rightarrow \text{stack}(X,Y)\]}
Semantics

- Meaning of expressions is a truth value assignment over the interpretation, I, e.g.,
  - const $\rightarrow$ object in the world,
  - function $\rightarrow$ object obtained by evaluating the function whose arguments were interpreted by I

- Example I: Logical sentence $\rightarrow$ \{T, F\} where
  - $I(\text{two}) = 2$, $I(\text{three}) = 3$, $I(\text{five}) = 6$
  - $I(\text{plus}(a, b)) = I(a) + I(b)$
  - $I(\text{equal}(a, b)) = \text{true}$ if $I(a) = I(b)$ else $= \text{false}$
  - $I(\text{true}) = \text{T}$ and $I(\text{false}) = \text{F}$

  This gives: $I(\text{equal}(\text{plus}(\text{two}, \text{three}), \text{five})) = \text{F}$

Logic as KR

- **Propositional Logic:** \{propositions + logical connectives\}
  - “AI is hard but it is interesting” = “p $\wedge$ q”, where
    - p = “AI is hard”, q = “AI is interesting”
  - “Not everyone likes AI”

- **Predicate Logic:** \{Terms + connectives + quantifiers + predicates\}
  - “Not everyone likes AI” = “$\neg \exists x (\text{likes}(x, \text{AI})$”
  - “She understands about 70% of the lecture”
  - “She understands almost everything. She is quite smart.”

Limitation of predicate logic:
- Can’t represent uncertain knowledge and fuzzy terms
- Truth assignment is of “syntax” not “semantic”
  - E.g., $(2+2 = 5) \Rightarrow (2+2+6 = 10)$ is true!
Use of Logic

- Mechanize “reasoning” involving
  Determining the “truth” of some statements (conclusion) based on the assumed “truth” of other statements (premises).

- Deal with “form” not “meaning”, e.g.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Propositions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob has full sight</td>
<td>P</td>
<td>T</td>
</tr>
<tr>
<td>Bob has one partial sight</td>
<td>Q</td>
<td>F</td>
</tr>
<tr>
<td>Bob is blind</td>
<td>R</td>
<td>F</td>
</tr>
</tbody>
</table>

  But \((P \equiv Q) \equiv R\) has value \((F \equiv F)\) which is \(T\)!

- The time to find valid interpretation of a formula of \(n\) propositions takes \(O(2^n)\)

Logic

- Logic as KR
  - Propositional Logic
  - Predicate Logic (predicate Calculus)

- Automated Reasoning
  - Logical inferences
  - Resolution and Theorem-proving
Logic as reasoning mechanisms

Examples of Inference rules

- Modus ponens \( \frac{p \rightarrow q, p}{q} \)
- And-introduction \( \frac{p, q}{p \land q} \)
- Universal-elimination \( \frac{\forall x (P(x))}{P(a)} \)
- Unit resolution \( \frac{p \lor q, \neg q}{p} \)
- Resolution \( \frac{p \lor q, \neg q}{p \lor r} \)

Resolution subsumes Modus ponens (exercise)
Resolution is more powerful than Modus ponens

Modus ponens

- Modus ponens is equivalent to Unit resolution

\[
\text{Since } p \land q = \neg p \lor q \\
\therefore \quad p \land q \lor p = \neg p \lor q \lor p \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
\]
Examples

Example: Given

1. P → Q
2. ~P → R
3. Q → R

Prove R

Rewrite:

1. ~P ∨ Q
2. P ∨ R
3. ~Q ∨ R

Applying resolution:

4. Q ∨ R (by resolving 1. and 2.)
5. R (by resolving 3. and 4.)

Unification

A process of determining whether two expressions can be made identical by appropriate "substitution" for their variables

- Unifier: \{α/β\} means substitute α for β (i.e., replace β by α)

  E.g., replace a variable A by
  - a constant, a \{a/A\}
  - another variable, B \{B/A\}
  - a function, f \{f(B)/A\}

- Examples:
  - `knows(bill, X)` and `knows(Y, Z)` → unified by \{bill/Y, X/Z\}
  - `foo(X, Y, a)` and `foo(b, f(b), Z)` → unified by \{b/X, f(b)/Y, a/Z\}
  - `knows(bill, X)` and `knows(bob, Z)` → not unifiable

Are `knows(bill, X)` and `knows(X, bob)` unifiable?
Unification (contd.)

- “occur checks” check to prevent x and p(X) from being unified together – not possible since
  
  unify (X, p(X)) by \{p(X)/X\} gives
  
  unify (p(X), p(p(X))) and so on. This leads to an infinite recursion

- There can be more than one unifiers - prefer the most general unifier (mgu) E.g., knows(bill, X) and knows(Y, Z) can be unified by
  
  \{bill/Y, X/Z\}
  
  \{bill/Y, mary/X, mary/Z\}
  
  \{bill/Y, f(a)/X, f(a)/Z\}
  
  ... but not \{Y/bill, X/Z\} and not \{bill/X, f(Z)/X, g(X)/Z\}

- See examples (e.g., ch. 2, Luger)

Skolemization

A process of eliminating existential quantifiers in an expression

Examples:

\[ \exists X \, p(X) \rightarrow p(a) \]

\[ \forall Y \exists X \, p(X, Y) \rightarrow \forall Y \, p(f(Y), Y) \]

\[ \forall Z \forall Y \exists X \, (p(X, Y) \rightarrow q(X)) \rightarrow \forall Z \forall Y \, (p(f(Z, Y), Y) \rightarrow q(f(Z, Y))) \]
**Inference Control**

Two Search Control Strategies:

- **Forward Chaining**
  
  ![Diagram showing forward chaining: Facts → Goal]

- **Backward Chaining**
  
  ![Diagram showing backward chaining: Fact → Goal]

**Automated Reasoning**

- **Deduction**
  - Reasons from general to specific
  - Uses logical inferences (Sound inferences)

- **Induction**
  - Reasons from specific to general
  - E.g., uses enumerative induction inference which is unsound
  
  \[
  p(a_1) \land p(a_2) \land \cdots \land p(a_n) \quad \forall x \ (\neg p(x))
  \]

- **Abduction**
  - Attempts to explain “conclusion” by assuming “premises” which is unsound
  
  \[
  \vdash q, q, \quad \frac{p \land q}{p}
  \]

  ![Diagram showing deduction and abduction: Explanation, Learning to describe, Diagnosis, Story understanding]
Completeness and Soundness

What are theorems?

- Not sound
  - The logic system might be able to prove some theorems that are not true in some interpretation, i.e., not valid
- Not complete
  - The logic system can’t prove all valid formulae, i.e., truths in the domain determined by the interpretation
- Predicate Logic is both sound and complete

More precisely....

A formula (expressions, sentences) is valid if it is true for every interpretation & assignment

Let T be a set of sentences and p be a sentence.
T logically implies p (T |= p) means p is valid if T is
p is provable from T (T |- p) means p is a provable theorem

Soundness: T |- p implies T |= p (every theorem must be valid)
Completeness: T |= p implies T |- p (every valid sentence must be provable)
**Theorem-proving**

**Problem:** Is \( p \models q \), i.e. can we derive a valid theorem \( q \) from a given set of axioms and formulae \( p \)?

**Why is it hard?**
- Many choices of inference rules (exponential search)
- The problem is semi-decidable, i.e., if \( p \models q \) is true, we can guarantee to show it, otherwise, we can’t.

\[
p \models p' \models \quad \text{How far shall we continue?}
\]

**We need:**
- A proof process that guarantees to stop when \( p \models q \)
- I.e., a complete proof procedure \( \rightarrow \) **Resolution Refutation**

**Resolution Refutation**

**Idea:**
- Negate the goal (theorem to be proved)
- Add it to the given set of formulae \( p \)
- Convert formulae to clausal forms, i.e., conjunction of disjunction of literals, e.g.,
  \[
  (\neg A \lor \neg B \lor C) \land (A \lor \neg D)
  \]
- Use resolution inference to resolve clauses
- If \([\ ]\), i.e., literal with value False, is resolved then answer YES, the goal is a theorem

**Example:** Is \{ \( (B \lor C) \), \( (\neg A \lor \neg B \lor C) \), \( (A \lor \neg D) \) \} \( \models (C \lor \neg D) \)?

\[
\begin{align*}
\text{Negate the goal: } & (\neg C \land D) \\
\neg A \lor \neg B \lor C & \land (A \lor \neg D) \\
B \lor C & \lor \neg B \lor C \lor D \\
\neg C \land D \lor C & \lor \neg D \\
\neg C & \lor \neg D \\
\text{The answer is YES}
\end{align*}
\]
Proof systems

- using Modus Ponens Inference is not complete

- using Resolution is still not complete BUT when used to produce refutation (contradiction), it is complete.
  I.e., Resolution is refutation-complete

Example: Given $\emptyset$. Prove $G \equiv P \lor \neg P$.

$G$ is valid but it can’t be proved by a proof system using Modus ponens or resolution (thus both proof systems are not complete).

Resolution refutation: $\neg G \equiv \neg P \land P$.
Resolve $\neg P$ and $P$ gives "False" = $\text{nil} = [\ ]$ indicating contradiction.
Thus, $G$ must be true.

Steps in Resolution Refutation

- Representing the problem in predicate logic
- Transform formulae into clauses [conjunction of disjunction of literals
  – atomic or negated atomic sentence, e.g.,
  $(B \lor C) \land (\neg A \lor \neg B \lor C) \land (A \lor \neg D) \ldots$ BUT NOT… $\forall Y \exists X p(X, Y) \rightarrow q(X)$ ]

  • Eliminate $\rightarrow$ (by rewriting $P \rightarrow Q$ by $\neg P \lor Q$)
  • Move $\sim$ down to atomic formula
  • Eliminate $\exists$ (by Skolem functions)
  • Renaming
  • Distribute $\lor$, separate conjunction parts & renaming
  • Eliminate $\forall$
- Construct resolution refutation tree: resolve clauses to find $[\ ]$
  • Use unification concept
**Skolemization**

A process of eliminating existential quantifiers in an expression

Examples:
\[ \exists X \ (p(X)) \rightarrow p(a) \]
\[ \forall Y \exists X \ p(X, Y) \rightarrow \forall Y \ p(f(Y), Y) \]
\[ \forall Z \forall Y \exists X \ (p(X, Y) \rightarrow q(X)) \rightarrow \forall Z \forall Y \ (p(f(Z, Y), Y) \rightarrow q(f(Z, Y))) \]

**Unification**

A process of determining whether two expressions can be made identical by appropriate "substitution" for their variables

- Unifier: \( \{\alpha/\beta\} \) means substitute \( \alpha \) for \( \beta \) (i.e., replace \( \beta \) by \( \alpha \))
- E.g., replace a variable A by
  - a constant, a \( \{a/A\} \)
  - another variable, B \( \{B/A\} \)
  - a function, f \( \{f(B)/A\} \)

- Examples:
  - knows(bill, X) and knows(Y, Z) \( \rightarrow \) unified by \( \{\text{bill}/Y, \ X/Z\} \)
  - foo(X, Y, a) and foo(b, f(b), Z) \( \rightarrow \) unified by \( \{\text{b}/X, \ f(b)/Y, \ a/Z\} \)
  - knows(bill, X) and knows(bob, Z) \( \rightarrow \) not unifiable
  - Are knows(bill, X) and knows(X, bob) unifiable?
Unification (contd.)

- **“occur checks”** check to prevent $x$ and $p(X)$ from being unified together – not possible since
  
  unify $(X, p(X))$ by $\{p(X)/X\}$ gives
  
  unify $(p(X), p(p(X)))$ and so on. This leads to an infinite recursion

- There can be more than one unifiers - prefer the most general unifier (mgu) E.g., knows(bill, $X$) and knows($Y$, $Z$) can be unified by
  
  $\{bill/Y, X/Z\}$
  
  $\{bill/Y, mary/X, mary/Z\}$
  
  $\{bill/Y, f(a)/X, f(a)/Z\}$
  
  ... but not $\{Y/bill, X/Z\}$ and not $\{bill/X, f(Z)/X, g(X)/Z\}$

- See examples (e.g., ch. 2, Luger)

More examples

**Given:** All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is not poor. Happy people have exciting life.

**Question:** Can anyone be found with an exciting life?

**Solution:** given in class.
Results from Resolution Refutation

• If there is a contradiction in axioms & ~theorem then the system is guaranteed to halt with a null clause [ ].

• If there is no contradiction between axioms & ~theorem then the system may not halt
  • If the system halts with no [ ] found then there is no contradiction between axioms & ~theorem.
  • But the theorem may be proved by other proof systems

• In other words, if a set of sentences is unsatisfiable then resolution will derive contradiction.

Logic Programming

• A programming language paradigm where logical assertions are viewed as programs, e.g. PROLOG. No concepts of “input” and “output” variables.

• In PROLOG
  • logical assertions are of form \( p_1 \land \ldots \land p_n \rightarrow q \) (Horn clauses)
  • Proof procedure of the goal = PROLOG program
  • Closed world assumption (any assertion that is not present is assumed to be false)

• Search engine is built in.
  • Programmers only specify logical assertions
  • Search strategies are fixed (e.g., PROLOG: backward chaining & DFS with backtracking)