Inference

- Inference: Sentence derived from others using inference methods
  - Inference (\(\vdash\)) vs. entailment (\(\models\))
  - Proof: Record of inference process, sequence of application of sound inference rules
- Sound inference method: no false conclusions
- Complete inference method: no missing conclusions (infers all that KB entails)

Inference Method

- Model checking
  - Time: \(O(2^n)\)
  - Space: \(O(n)\)
  - Sound and complete
- Forward Chaining
- Backward Chaining

Inference Rules

- Modus Ponens:
- Modus Tollens:
- And-elim
- Biconditional-elim:
- Contrapositive:
- Addition:
- Adjunction:

Other useful rules for inference

- DeMorgan’s
  - \(\neg (a \lor b) \equiv \neg a \land \neg b\)
  - \(\neg (a \land b) \equiv \neg a \lor \neg b\)
- Distribution
  - \((a \land (b \lor c)) \equiv ((a \land b) \lor (a \land c))\)
  - \((a \lor (b \land c)) \equiv ((a \lor b) \land (a \lor c))\)
Resolution
- Unit resolution
  \[ \alpha \lor \beta, \neg \beta \]
- Full resolution
  \[ \left( c \lor a_1 \lor a_2 \lor \ldots \lor a_j \right) \land \left( \neg c \lor b_1 \lor b_2 \lor \ldots \lor b_k \right) \]
  \[ a_1 \lor a_2 \lor \ldots \lor a_j \lor b_1 \lor b_2 \lor \ldots \lor b_k \]
- Can be used as a single rule for a complete inference algorithm

CNF
- Conjunctive normal form
  - Conjunction of disjunction literals
  - Especially useful for resolution
  - k-CNF: has k literals per clause
    \[ (l_1 \lor \ldots \lor l_j) \land (l_2 \lor \ldots \lor l_j) \land \ldots \land (l_m \lor \ldots \lor l_j) \]
- Every PL can be transformed into 3-CNF
- Conversion

Refutation proof
- 1. Convert all sentences into CNF
- 2. Negate the sentence you want to prove and insert the negated sentence into knowledge base
- 3. Use resolution rules to derive contradiction

First Order Logic
- Object
  - Predicates: test property of object(s)
    - IsClean(x)
    - With 0 argument is a proposition
  - Functions: represent relationships
    - MotherOf(x)
    - With 0 argument is a constant object
  - Equality: term1=term2
- Variables
  - Quantifiers
    - Universal quantifier \( \forall \)
      - For all
      - Pair with \( \exists \)
      - Universal instantiation (Replace var. with ground term)
    - Existential quantifier \( \exists \)
      - There exists
      - Pair with \( \forall \)
      - Existential instantiation (Replace var. with Skolem constant)

Herbrand’s Theorem
- PL converting to FOL
  - If a sentence entails KB, we can prove by propositionalizing KB written in FOL in finite time (proof only involves finite subset of propositionalized KB).
  - If a sentence does not entail KB, infinite loop (due to function) is possible.
- Semi-decidable
Unification

- A process of finding substitution to make things look the same on demand
- Failure occurs when
  - Structure mismatch: \( Q(\ldots), P(\ldots) \)
  - Ground terms conflict: \( P(\text{Lion}), P(\text{Tiger}) \)
  - A variable bound to different ground terms
    - \( x/\text{Lion}, x/\text{Tiger} \)
  - Fails “occur-check”
    - \( Q(y), Q(f(y)) \)

Datalog vs. Prolog

- **Datalog**
  - First order definite clauses
    - Disjunction of literals, exactly one positive term
    - Similar to horn clauses: \( P \vee Q \Rightarrow R \)
  - No function allowed
  - Forward chaining
- **Prolog**
  - Datalog+function
  - Depth-first backward chaining
  - Loop possible

Proof

- **Forward chaining**
  - Inference occurs when sentences inserted to KB (TELL)
  - Why don’t we always use forward chaining
- **Backward chaining**
  - Inference occurs when you ASK KB
- **Examples**
  - Refer to Discussion 6 slides

Planning

- **Situation calculus**
  - Covered in review session
- **Progression**
  - Procedure in page 10 of lecture 13 slides
- **Regression**
  - Procedure in page 11 of lecture 13 slides
- **Graph planning** – concrete example in Lecture 14

Probability

- **Formulas:**
  - An axiom: \( P(a \lor b) = P(a) + P(b) - P(a \land b) \)
  - What if violating this p490
  - Product rule: \( P(a \land b) = P(a | b) \cdot P(b) \)
  - Bayes rule: \( P(a | b) = P(b | a) \cdot P(a) / P(b) = P(b \land a) / P(b) \)
- **Concepts:**
  - Independence: \( P(a | b) = P(a) \)
  - Implies: \( P(a \land b) = P(a)P(b) \)
  - Conditionally independence:
    - \( a \land b \) conditionally independence given \( c \)
    - \( P(a \lor b | c) = P(a | c)P(b | c) \)
    - \( P(a | b, c) = P(a | c) \)

Causal vs. Diagnostic

- **Causal:** \( P(\text{effect} | \text{cause}) \)
- **Diagnostic:** \( P(\text{cause} | \text{effect}) \)

- In Bayes rule: \( P(H | e) = P(e | H) \cdot P(H) / P(e) \)
  - \( P(H) \): prior prob. of hypothesis
  - \( P(H | e) \): posterior
  - \( P(e | H) \): likelihood
  - \( P(e) \): prior prob. of evidence
  - \( P(\text{disease} | \text{positive test}) \)
Joint Probability Table
- Example in lecture
- All values sum to one
- Complete
- \( P(H_1 \land \lnot H_2) = 0.21 \) (joint probability)
- \( P(H_1) = 0.49 + 0.21 = 0.7 \) (marginalization)

Conditional Probability Table
- Evidence involved
- \( P(A \mid B) + P(\lnot A \mid B) = 1 \)
- Product rule \( P(a \land b) = P(a \mid b)P(b) \)
relates joint and conditional probability table

Bayesian Network
- Independence of variables represented by graph
  - Exploiting independence avoids us from writing joint distribution, which is not realistic for most cases
  - Consider \( P(A, B, C, D) = P(A \mid B, C, D)P(B \mid C, D)P(C \mid D)P(D) \)
  - If when discover \( C, D \) are independent and \( A, B \) are conditional independent given \( C \), we have
    \[ P(A \mid C, D)P(B \mid C, D)P(C)P(D) \]

Independence Given a Bayes Network
- Head-to-tail
  - \( A, B \) dependent
  - Given knowledge \( C \)
    - \( A, B \) independent

Independence Given a Bayes Network
- Tail-to-tail
  - \( A, B \) dependent
  - Given knowledge \( C \)
    - \( A, B \) independent

Independence Given a Bayes Network
- Head-to-head
  - \( A, B \) independent
  - Given knowledge \( C \)
    - \( A, B \) dependent
Independence Given a Bayes Network

- Head-to-head

- Explaining away
  - Knowing A is true makes B unlikely
  - Knowing B is true makes A unlikely

- Given knowledge of C

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Knowing D, a decedent of C, also makes A,B dependent (because C and D are dependent, knowing D gives us knowledge about C)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

Reconstructing a graph model

- MJAEB

- AMEJB

<table>
<thead>
<tr>
<th>Burglary</th>
<th>Earthquake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td></td>
</tr>
<tr>
<td>JohnCalls</td>
<td>MaryCalls</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Burglary</th>
<th>Earthquake</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td></td>
</tr>
<tr>
<td>JohnCalls</td>
<td>MaryCalls</td>
</tr>
<tr>
<td>MJAEB</td>
<td></td>
</tr>
<tr>
<td>AMEJB</td>
<td></td>
</tr>
</tbody>
</table>