Today

• Knowledge Representation and Reasoning (R&N Ch 7-9)

Tuesday, March 21

• Prelim, Statler Auditorium

Thursday, March 23

• Knowledge Representation and Reasoning (R&N Ch 8-9)
Resolution Example: Rules

KB:

P \Rightarrow Q
L \land M \Rightarrow P
B \land L \Rightarrow M
A \land P \Rightarrow L
A \land B \Rightarrow L
A
B

KB \models P \land Q?
Resolution Example: Rules

$$\text{KB:} \quad \begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A & \\
B &
\end{align*}$$

$$\text{Is the following CNF unsatisfiable?}$$

$$\begin{align*}
(\neg P \lor Q) \land \\
(\neg L \lor \neg M \lor P) \land \\
(\neg B \lor \neg L \lor M) \land \\
(\neg A \lor \neg P \lor L) \land \\
(\neg A \lor \neg B \lor L) \land \\
A \land \\
B \land \\
(\neg P \lor \neg Q)
\end{align*}$$

$$\text{KB} \models P \land Q?$$
Horn Clause Logic

KB:

P \implies Q
L \land M \implies P
B \land L \implies M
A \land P \implies L
A \land B \implies L
A
B

KB \models P \land Q?

Is the following CNF unsatisfiable?

\neg (P \lor Q) \land
\neg (L \lor \neg M \lor P) \land
\neg (B \lor \neg L \lor M) \land
\neg (A \lor \neg P \lor L) \land
\neg (A \lor \neg B \lor L) \land
A \land
B \land
\neg (P \lor \neg Q)
Horn Clause Logic

• Horn clauses:
  • A sentence in CNF where each clause (disjunction) has at most one unnegated literal
  • Typically arises when KB is a set of implications whose antecedents are conjunctions of unnegated literals

• Inference methods:
  • Forward chaining: Start from unnegated literal facts, add new facts from rules whose antecedents are true, apply recursively
Horn Clause Logic

Forward chaining example:

B [7]

KB:

1. P \Rightarrow Q
2. L \land M \Rightarrow P
3. B \land L \Rightarrow M
4. A \land P \Rightarrow L
5. A \land B \Rightarrow L
6. A
7. B
Horn Clause Logic

Forward chaining example:

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>[7]</td>
</tr>
<tr>
<td>L</td>
<td>[5]</td>
</tr>
</tbody>
</table>

KB:

1. $P \Rightarrow Q$
2. $L \land M \Rightarrow P$
3. $B \land L \Rightarrow M$
4. $A \land P \Rightarrow L$
5. $A \land B \Rightarrow L$
6. A
7. B
Horn Clause Logic

Forward chaining example:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

KB:

1. $P \Rightarrow Q$
2. $L \land M \Rightarrow P$
3. $B \land L \Rightarrow M$
4. $A \land P \Rightarrow L$
5. $A \land B \Rightarrow L$
6. $A$
7. $B$
Horn Clause Logic

Forward chaining example:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>2</td>
</tr>
</tbody>
</table>

KB:

1. $P \Rightarrow Q$
2. $L \land M \Rightarrow P$
3. $B \land L \Rightarrow M$
4. $A \land P \Rightarrow L$
5. $A \land B \Rightarrow L$
6. $A$
7. $B$
## Horn Clause Logic

### Forward chaining example:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Rule Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
</tr>
</tbody>
</table>

### KB:

1. \( P \Rightarrow Q \)
2. \( L \land M \Rightarrow P \)
3. \( B \land L \Rightarrow M \)
4. \( A \land P \Rightarrow L \)
5. \( A \land B \Rightarrow L \)
6. A
7. B
Horn Clause Logic

• Which facts to combine with which rules?

• This is a search problem:
  • States: Sets of facts
  • Operators: The set of rules
    • If the given state contains all the facts necessary for a rule, add the rule’s consequence to the set of facts to create a new state
  • Initial state: Starting facts
  • Goal: One or more facts that are being asked about
Horn Clause Logic

• Example:
  • Initial state: \{A,B\}
  • Operators: \{A \land B \Rightarrow L, A \land P \Rightarrow L, \ldots\}
  • Result of applying \(A \land B \Rightarrow L\) to initial state: \{A,B,L\}
  • Goal state: A state containing P and Q
Horn Clause Logic

• Horn clauses:
  • A sentence in CNF where each clause (disjunction) has at most one unnegated literal
  • Typically arises when KB is a set of implications whose antecedents are conjunctions of unnegated literals

• Inference methods:
  • Forward chaining: Start from unnegated literal facts, add new facts from rules whose antecedents are true, apply recursively
Horn Clause Logic

• Horn clauses:
  • A sentence in CNF where each clause (disjunction) has at most one unnegated literal
  • Typically arises when KB is a set of implications whose antecedents are conjunctions of unnegated literals

• Inference methods:
  • Forward chaining: Start from unnegated literal facts, add new facts from rules whose antecedents are true, apply recursively
  • Backward chaining: Start from goal, find rule that concludes it, apply recursively to antecedents (this is the core of what the Prolog programming language does)
Horn Clause Logic

• Horn clauses:
  • A sentence in CNF where each clause (disjunction) has at most one unnegated literal
  • Typically arises when KB is a set of implications whose antecedents are conjunctions of unnegated literals

• Inference methods:
  • Forward chaining: Start from unnegated literal facts, add new facts from rules whose antecedents are true, apply recursively
  • Backward chaining: Start from goal, find rule that concludes it, apply recursively to antecedents (this is the core of what the Prolog programming language does)
  • There is a linear time algorithm for satisfiability of Horn clauses
Satisfiability

At one extreme:
- Tautology: Sentence is true no matter what truth assignments are given to the propositional variables

At the other extreme:
- Unsatisfiable: The sentence is never true no matter what truth assignments are given to the propositional variables

Satisfiability:
- Is the sentence true for any truth assignment?
- What truth assignment makes the sentence true?
Example: N-Queens

• Encode in propositional logic
  • $Q_{ij}$: There is a queen in position $(i,j)$
  • One queen in each row, column, diagonal:
    • $(Q_{i1} \lor Q_{i2} \lor \ldots \lor Q_{iN})$ for $i \in \{1,\ldots,N\}$
    • $(Q_{1j} \lor Q_{2j} \lor \ldots \lor Q_{Nj})$ for $i \in \{1,\ldots,N\}$
    • $(Q_{11} \lor Q_{22} \lor \ldots \lor Q_{NN})$ etc. for each diagonal
  • No two attacking queens:
    • $\neg (Q_{ab} \land Q_{ac})$ for all $b \neq c$
    • $\neg (Q_{ab} \land Q_{db})$ for all $a \neq d$
    • $\neg (Q_{ij} \land Q_{kl})$ for all $k=i\pm m$, $l=j\pm m$, $m \neq 0$
    • $\neg (Q_{ij} \land Q_{kl})$ for all $i+j = k+l$, $i \neq k$
Example: N-Queens

• Unsatisfiable: Means there is no placement of the queens (N=2, 3)

• Satisfiable: Means there is a placement of the queens (N > 3)

• Don’t just want to be told there is a placement, want an actual placement!
Searching for Truth Assignments

• Brute-force search:
  • Assign a variable True or False
  • Is any clause violated?
    • If not, go on to another variable
    • If yes, backtrack to most recent variable assignment and flip it
      • If both options have been tried and failed, backtrack to the next most recent assignment
  • Complete
• Davis-Putnam: Backtracking exploiting some additional observations that simplify the problem in certain cases
• Further efficiency: Grab bag of clever ideas
Searching for Truth Assignments

• Hill climbing: Surprisingly effective
  • Pick random assignment
  • If it doesn’t satisfy the sentence, flip whichever one most increases the number of satisfied terms and repeat
  • If there are none, restart
First Order Logic

- **Sentence** → AtomicSentence | ComplexSentence
- **AtomicSentence** → Predicate | Predicate(Arguments) | Term = Term
- **Arguments** → Term | Term,Arguments
- **Term** → Function(Arguments) | Constant | Variable
- **ComplexSentence** → (Sentence) | [Sentence] | ¬ Sentence
  - Sentence ∧ Sentence | Sentence ∨ Sentence
  - Sentence ⇒ Sentence | Quantifiers Sentence
- **Quantifiers** → Quantifier Variables | Quantifier Variables Quantifiers
- **Quantifier** → ∀ | ∃
- **Constant/Predicate/Function** → <strings starting with upper case letters>
- **Variables** → Variable | Variable,Variables
- **Variable** → <strings comprised of lower case letters>
Examples

• Mother(Alice,Charlie)
• Father(Bob,Charlie)
Examples

- Mother(Alice, Charlie)
- Father(Bob, Charlie)
- $\forall x, y \text{ Mother}(x, y) \Rightarrow \text{Parent}(x, y)$
- $\forall x, y \text{ Father}(x, y) \Rightarrow \text{Parent}(x, y)$
Examples

• Mother(Alice, Charlie)
• Father(Bob, Charlie)
• $\forall \ x, y \ Mother(x, y) \Rightarrow Parent(x, y)$
• $\forall \ x, y \ Father(x, y) \Rightarrow Parent(x, y)$
• $\forall \ x, y \ Parent(x, y) \Rightarrow Ancestor(x, y)$
Examples

• Mother(Alice,Charlie)
• Father(Bob,Charlie)
• $\forall x, y \; Mother(x,y) \Rightarrow Parent(x,y)$
• $\forall x, y \; Father(x,y) \Rightarrow Parent(x,y)$
• $\forall x, y \; Parent(x,y) \Rightarrow Ancestor(x,y)$
• $\forall x, y, z \; Ancestor(x,y) \land Ancestor(y,z) \Rightarrow Ancestor(x,z)$
Examples

• Mother(Alice,Charlie)
• Father(Bob,Charlie)
• \( \forall x,y \) Mother\((x,y) \Rightarrow Parent(x,y) \)
• \( \forall x,y \) Father\((x,y) \Rightarrow Parent(x,y) \)
• \( \forall x,y \) Parent\((x,y) \Rightarrow Ancestor(x,y) \)
• \( \forall x,y,z \) Ancestor\((x,y) \land Ancestor(y,z) \Rightarrow Ancestor(x,z) \)
• \( \forall x \exists y \) Mother\((y,x) \)
• \( \forall x \exists y \) Father\((y,x) \)
Examples

- Mother(Alice, Charlie)
- Father(Bob, Charlie)
- $\forall x, y \text{ Mother}(x, y) \Rightarrow \text{Parent}(x, y)$
- $\forall x, y \text{ Father}(x, y) \Rightarrow \text{Parent}(x, y)$
- $\forall x, y \text{ Parent}(x, y) \Rightarrow \text{Ancestor}(x, y)$
- $\forall x, y, z \text{ Ancestor}(x, y) \land \text{Ancestor}(y, z) \Rightarrow \text{Ancestor}(x, z)$
- $\forall x \exists y \text{ Mother}(y, x)$
- $\forall x \exists y \text{ Father}(y, x)$
- $\exists x \not\exists y \text{ Mother}(x, y)$
Inference in First Order Logic

• First-order Horn clauses are an analogous subset of first-order logic
  • They have analogs of forward chaining and backward chaining
  • Prolog is a programming language based on backward chaining on first-order
    Horn clauses

• There is a conjunctive normal form for First-Order Logic
• There is a resolution inference rule for First-Order Logic
CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere [no more “$\Rightarrow$”]

Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$ [push “¬” inward until you can’t any more]

Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$ [“¬” only before prop symbols]

Replace $\neg \neg \alpha$ with $\alpha$ [“¬” only before prop symbols]

Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
CNF

Replace $\alpha \implies \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$

Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Replace $\neg \forall x$ with $\exists x$ $\neg$
Replace $\neg \exists x$ with $\forall x$ $\neg$
Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
Replace \( \alpha \implies \beta \) with \( \neg \alpha \lor \beta \) everywhere

Replace \( \neg (\alpha \lor \beta) \) with \( \neg \alpha \land \neg \beta \)

Replace \( \neg (\alpha \land \beta) \) with \( \neg \alpha \lor \neg \beta \)

Replace \( \neg \neg \alpha \) with \( \alpha \)

Replace \( \neg \forall x \) with \( \exists x \neg \)

Replace \( \neg \exists x \) with \( \forall x \neg \)

Distribute \( \lor \) over \( \land \): Rewrite \( \alpha \lor (\beta \land \gamma) \) as \( (\alpha \lor \beta) \land (\alpha \lor \gamma) \)
CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Replace $\neg \forall x$ with $\exists x \neg$
Replace $\neg \exists x$ with $\forall x \neg$

Standardize variables: If the same variable name is used in multiple places with multiple quantifiers, rename them so there is no duplication

$\forall x P(x) \lor \forall x Q(x)$ becomes $\forall x P(x) \lor \forall z Q(z)$ [where $z$ is new]

Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
CNF

Replace \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \) everywhere
Replace \( \neg (\alpha \lor \beta) \) with \( \neg \alpha \land \neg \beta \)
Replace \( \neg (\alpha \land \beta) \) with \( \neg \alpha \lor \neg \beta \)
Replace \( \neg \neg \alpha \) with \( \alpha \)
Replace \( \neg \forall x \) with \( \exists x \neg \)
Replace \( \neg \exists x \) with \( \forall x \neg \)

Standardize variables: If the same variable name is used in multiple places with multiple quantifiers, rename them so there is no duplication
\[
\forall x P(x) \lor \forall x Q(x) \text{ becomes } \forall x P(x) \lor \forall z Q(z) \text{ [where } z \text{ is new]}\]

Distribute \( \lor \) over \( \land \): Rewrite \( \alpha \lor (\beta \land \gamma) \) as \( (\alpha \lor \beta) \land (\alpha \lor \gamma) \)
CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Replace $\neg \forall x$ with $\exists x \neg$
Replace $\neg \exists x$ with $\forall x \neg$

Standardize variables: If the same variable name is used in multiple places with multiple quantifiers, rename them so there is no duplication

$\forall x P(x) \lor \forall x Q(x)$ becomes $\forall x P(x) \lor \forall z Q(z)$ [where $z$ is new]

Skolemization

Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Replace $\neg \forall x$ with $\exists x \neg$
Replace $\neg \exists x$ with $\forall x \neg$
Standardize variables: If the same variable name is used in multiple places with multiple quantifiers, rename them so there is no duplication
\[ \forall x \ P(x) \lor \forall x \ Q(x) \] becomes \[ \forall x \ P(x) \lor \forall z \ Q(z) \] [where $z$ is new]

Skolemization

Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
CNF

Replace $\alpha \implies \beta$ with $\neg \alpha \lor \beta$ everywhere
Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$
Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$
Replace $\neg \neg \alpha$ with $\alpha$
Replace $\neg \forall x$ with $\exists x \neg$
Replace $\neg \exists x$ with $\forall x \neg$

Standardize variables: If the same variable name is used in multiple places with multiple quantifiers, rename them so there is no duplication

$\forall x P(x) \lor \forall x Q(x)$ becomes $\forall x P(x) \lor \forall z Q(z)$ [where $z$ is new]

Skolemization
Drop universal quantifiers
Distribute $\lor$ over $\land$: Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
Skolemization

• $\exists x \ P(x)$ becomes $P(A)$ where $A$ is a constant that is new
Skolemization

- $\exists x \ P(x)$ becomes $P(A)$ where $A$ is a new constant

- $\forall x, y \ \exists z \ P(z)$ becomes $P(F(x,y))$ where $F$ is a new function
Skolemization

- $\exists x \ P(x)$ becomes $P(A)$ where $A$ is a new constant

- $\forall x, y \ \exists z \ P(z)$ becomes $P(F(x,y))$ where $F$ is a new function

- Replace existentially quantified variables with Skolem functions – new functions whose arguments are all universally quantified variables of higher scope
Resolution in First Order Logic

Recall:

$$(\ell \lor \alpha_1 \lor \ldots \lor \alpha_k)$$

$$(\neg \ell \lor \beta_1 \lor \ldots \lor \beta_n)$$

$$(\alpha_1 \lor \ldots \lor \alpha_k \lor \beta_1 \lor \ldots \lor \beta_n)$$

where each $\alpha_i$ and $\beta_j$ are propositional symbols or their negations and $\ell$ is a propositional symbol.
Resolution in First Order Logic

Here:

\[(\ell \lor \alpha_1 \lor ... \lor \alpha_k)\]  
\[(-\ell' \lor \beta_1 \lor ... \lor \beta_n)\]  

\[\text{Subst}(\varepsilon, \alpha_1 \lor ... \lor \alpha_k \lor \beta_1 \lor ... \lor \beta_n)\]

where \(\varepsilon = \text{Unify}(\ell, -\ell')\)

[\(\ell\) and \(-\ell'\) can be made identical by substituting variables with terms]  
[the necessary substitution]