CS 4700: Foundations of Artificial Intelligence

Fall 2017

Instructor: Prof. Haym Hirsh

Lecture 14

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:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

$KB \vDash P \wedge Q$?

Resolution Example: Rules

KB:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

$KB \models P \wedge Q$?

Is the following CNF unsatisfiable?

$(\neg P \vee Q) \wedge$

$(\neg L \vee \neg M \vee P) \wedge$

$(\neg B \vee \neg L \vee M) \wedge$

$(\neg A \vee \neg P \vee L) \wedge$

$(\neg A \vee \neg B \vee L) \wedge$

$A \wedge$

$B \wedge$

$(\neg P \vee \neg Q)$

Horn Clause Logic

KB:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

$KB \models P \wedge Q$?

Is the following CNF unsatisfiable?

$(\neg P \vee Q) \wedge$

$(\neg L \vee \neg M \vee P) \wedge$

$(\neg B \vee \neg L \vee M) \wedge$

$(\neg A \vee \neg P \vee L) \wedge$

$(\neg A \vee \neg B \vee L) \wedge$

$A \wedge$

$B \wedge$

$(\neg P \vee \neg Q)$

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:

A [6]

B [7]

KB:

1. $P \Rightarrow Q$
2. $L \wedge M \Rightarrow P$
3. $B \wedge L \Rightarrow M$
4. $A \wedge P \Rightarrow L$
5. $A \wedge B \Rightarrow L$
6. A
7. B

Horn Clause Logic

Forward chaining example:

A	[6]
B	[7]
L	[5]

KB:

1. $P \Rightarrow Q$
2. $L \wedge M \Rightarrow P$
3. $B \wedge L \Rightarrow M$
4. $A \wedge P \Rightarrow L$
5. $A \wedge B \Rightarrow L$
6. A
7. B

Horn Clause Logic

Forward chaining example:

A	[6]
B	[7]
L	[5]
M	[3]

KB:

1. $P \Rightarrow Q$
2. $L \wedge M \Rightarrow P$
3. $B \wedge L \Rightarrow M$
4. $A \wedge P \Rightarrow L$
5. $A \wedge B \Rightarrow L$
6. A
7. B

Horn Clause Logic

Forward chaining example:

A	[6]
B	[7]
L	[5]
M	[3]
P	[2]

KB:

1. $P \Rightarrow Q$
2. $L \wedge M \Rightarrow P$
3. $B \wedge L \Rightarrow M$
4. $A \wedge P \Rightarrow L$
5. $A \wedge B \Rightarrow L$
6. A
7. B

Horn Clause Logic

Forward chaining example:

A	[6]
B	[7]
L	[5]
M	[3]
P	[2]
Q	[1]

KB:

1. $P \Rightarrow Q$
2. $L \wedge M \Rightarrow P$
3. $B \wedge L \Rightarrow M$
4. $A \wedge P \Rightarrow L$
5. $A \wedge B \Rightarrow L$
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: $\{A \wedge B \Rightarrow L,\ A \wedge P \Rightarrow L, \ldots\}$
  - Result of applying $A \wedge B \Rightarrow L$ 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
  - Qij: There is a queen in position (i,j)
  - One queen in each row, column, diagonal:
    - $(Qi1 \lor Qi2 \lor \ldots \lor QiN)$ for $i \in \{1,\ldots,N\}$
    - $(Q1j \lor Q2j \lor \ldots \lor QNj)$ for $i \in \{1,\ldots,N\}$
    - $(Q11 \lor Q22 \lor \ldots \lor QNN)$ etc. for each diagonal
  - No two attacking queens:
    - $\neg (Qab \land Qac)$ for all $b \neq c$
    - $\neg (Qab \land Qdb)$ for all $a \neq d$
    - $\neg (Qij \land Qkl)$ for all $k=i\pm m$, $l=j\pm m$, $m \neq 0$
    - $\neg (Qij \land Qkl)$ 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] | $\neg$ Sentence
  | Sentence $\wedge$ Sentence | Sentence $\vee$ Sentence
  | Sentence $\Rightarrow$ Sentence | Quantifiers Sentence
- Quantifiers → Quantifier Variables | Quantifier Variables Quantifiers
- Quantifier → $\forall$ | $\exists$
- 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 Mother(x,y) $\Rightarrow$ Parent(x,y)
- $\forall$ x,y Father(x,y) $\Rightarrow$ 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) $\wedge$ 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) $\wedge$ 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 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) $\wedge$ Ancestor(y,z) $\Rightarrow$ Ancestor(x,z)
- $\forall$ x $\exists$ y Mother(y,x)
- $\forall$ x $\exists$ y Father(y,x)
- $\exists$ x $\neg$ $\exists$ y 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 $\neg \alpha \vee \beta$ everywhere [no more "$\Rightarrow$"]

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$ [push "$\neg$" inward until

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$ you can't any more]

Replace $\neg \neg \alpha$ with α ["$\neg$" only before prop symbols]

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere
Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$
Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$
Replace $\neg \neg \alpha$ with α

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

Replace $\neg \forall x$ with $\exists x \neg$

Replace $\neg \exists x$ with $\forall x \neg$

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

Replace $\neg \forall x$ with $\exists x \neg$

Replace $\neg \exists x$ with $\forall x \neg$

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

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

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

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

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

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

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

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

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

Skolemization

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

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

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

Skolemization

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

CNF

Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$ everywhere

Replace $\neg (\alpha \vee \beta)$ with $\neg \alpha \wedge \neg \beta$

Replace $\neg (\alpha \wedge \beta)$ with $\neg \alpha \vee \neg \beta$

Replace $\neg \neg \alpha$ with α

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

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

Skolemization

Drop universal quantifiers

Distribute $\vee$ over $\wedge$: Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

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:

$$\frac{(\ell \vee \alpha_1 \vee \ldots \vee \alpha_k) \qquad (\neg \ell \vee \beta_1 \vee \ldots \vee \beta_n)}{(\alpha_1 \vee \ldots \vee \alpha_k \vee \beta_1 \vee \ldots \vee \beta_n)}$$

where each α_i and β_j are propositional symbols or their negations and ℓ is a propositional symbol

Resolution in First Order Logic

Here:

$(\ell \vee \alpha_1 \vee \ldots \vee \alpha_k)$ [ℓ and $\neg\ell'$ can be made identical by

$\underline{(\neg\ell' \vee \beta_1 \vee \ldots \vee \beta_n)}$ substituting variables with terms]

$\text{Subst}(\theta, \alpha_1 \vee \ldots \vee \alpha_k \vee \beta_1 \vee \ldots \vee \beta_n)$ [the necessary substitution]

where $\theta = \text{Unify}(\ell, \neg\ell')$