First-order Logic

CS 486/686

Nov 19, 2009

University of Waterloo

Outline

- First-order logic
  - Syntax and semantics
- Inference
  - Propositionalization with ground inference
  - Lifted resolution

Logic

- General language
  - Can encode any kind of deterministic and discrete knowledge
  - Well defined syntax and semantics
  - Knowledge base: store domain knowledge
- General algorithms
  - Search by inference
  - Can infirm or confirm conclusions

Syntax of a very simple logic

- Sentence → AtomicSentence | ComplexSentence
- AtomicSentence → True | False | Symbol
- Symbol → P | Q | R | …
- ComplexSentence → ¬Sentence
  | (Sentence ∧ Sentence)
  | (Sentence ∨ Sentence)
  | (Sentence ⇒ Sentence)
  | (Sentence ⇔ Sentence)

  Symbols are binary variables

Logical connector semantics

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Recall the Street Puzzle

1	2	3	4	5

N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house
The Spaniard has a Dog
The Japanese is a Painter
The Italian drinks Tea
The Norwegian lives in the first house on the left
The owner of the Green house drinks Coffee
The Green house is on the right of the White house
The Sculptor breeds Snails
The Diplomat lives in the Yellow house
The owner of the middle house drinks Milk
The Norwegian lives next door to the Blue house
The Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

Who owns the Zebra?
Who drinks Water?

Example – Street Puzzle

- Symbols: V_{in}, V_{ic}, V_{id}, V_{ij}, V_{ia}

  $i \in \{1,2,3,4,5\}$
  $n \in \{\text{English, Spaniard, Japanese, Italian, Norwegian}\}$
  $c \in \{\text{Red, Green, White, Yellow, Blue}\}$
  $d \in \{\text{Tea, Coffee, Milk, Fruit-juice, Water}\}$
  $j \in \{\text{Painter, Sculptor, Diplomat, Violinist, Doctor}\}$
  $a \in \{\text{Dog, Snails, Fox, Horse, Zebra}\}$

- Example:
  V_{2red} = true: the 2nd house is red
  V_{2red} = false: the 2nd house is not red

Example – Street Puzzle

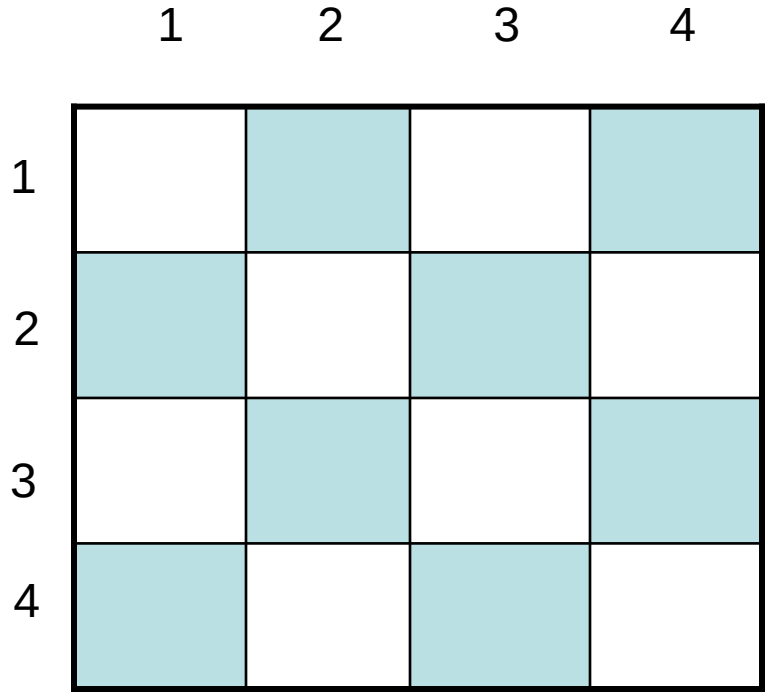
- Constraint: the Spaniard has a dog

$$(V_{1spaniard} \Leftrightarrow V_{1dog}) \wedge (V_{2spaniard} \Leftrightarrow V_{2dog}) \wedge$$
$$(V_{3spaniard} \Leftrightarrow V_{3dog}) \wedge (V_{4spaniard} \Leftrightarrow V_{4dog}) \wedge$$
$$(V_{5spaniard} \Leftrightarrow V_{5dog})$$

- Constraint: the green house is on the immediate left of the red house

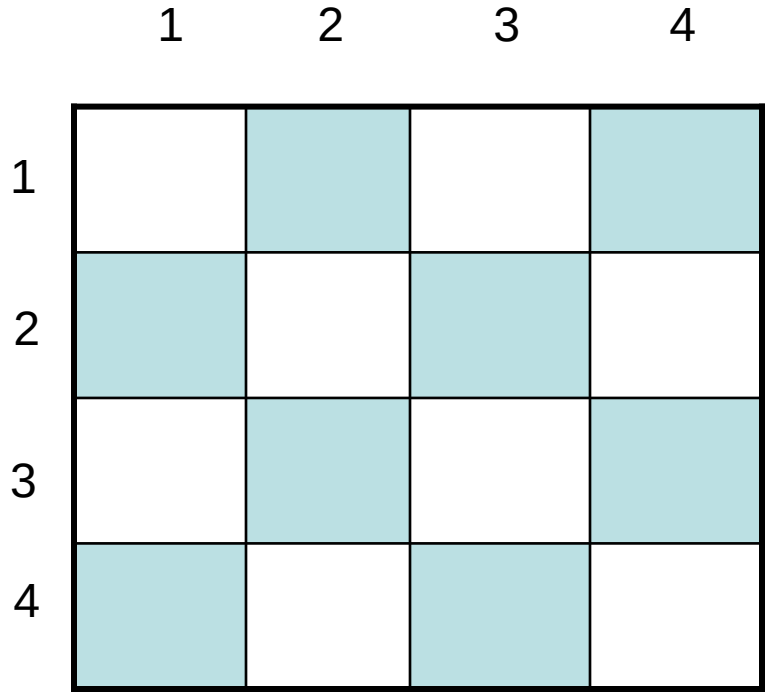
$$\neg V_{1red} \wedge (V_{1green} \Leftrightarrow V_{2red}) \wedge (V_{2green} \Leftrightarrow V_{3red}) \wedge$$
$$(V_{3green} \Leftrightarrow V_{4red}) \wedge (V_{4green} \Leftrightarrow V_{5red}) \wedge \neg V_{5green}$$

Example: 4-Queens problem



- Symbols: Q_{ij} $i,j \in \{1,2,3,4\}$
- Q_{ij} = true: queen at location (i,j)
- Q_{ij} = false: no queen at location (i,j)

Example: 4-Queens problem



- At least one queen in row 1:

$$Q_{11} \vee Q_{12} \vee Q_{13} \vee Q_{14}$$

- At most one queen in row 1:

$$Q_{11} \Rightarrow \neg Q_{12} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{12} \Rightarrow \neg Q_{11} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{13} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{14}$$
$$Q_{14} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{13}$$

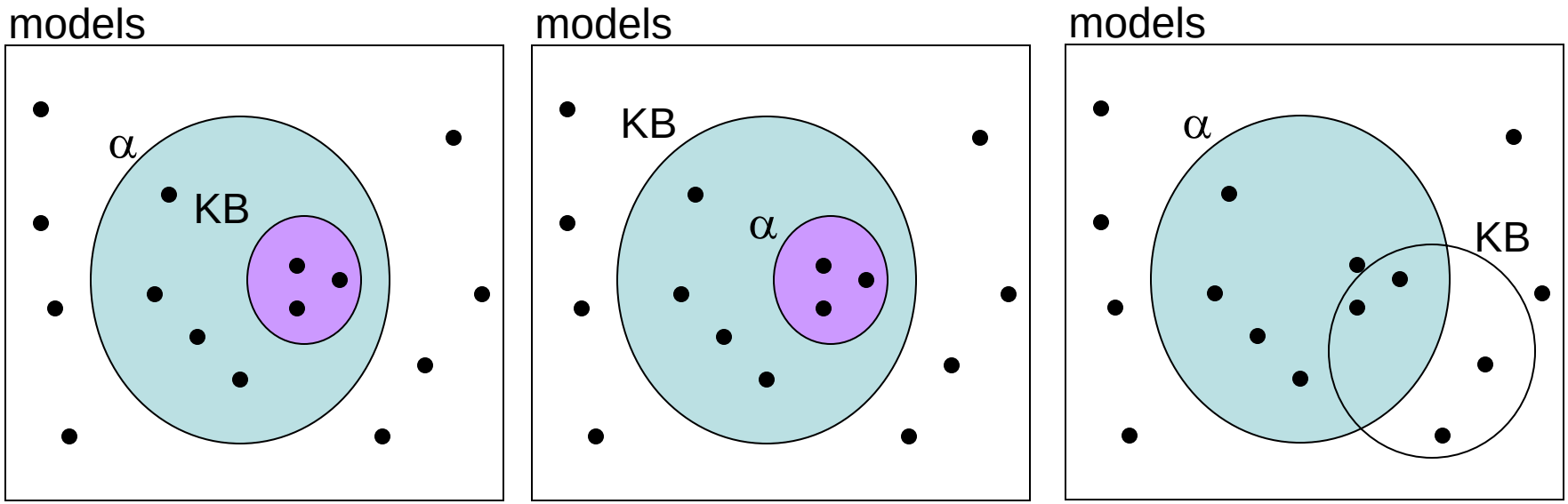
Knowledge Base (KB)

- Knowledge base
  - Encode problem description and any knowledge that could help solve the problem
  - Database of facts and constraints
  - Possible language: propositional logic

- Entailment
  - What can we derive/conclude from KB?
  - KB |= α: α follows from KB

Semantics

- Entailment: KB |= α
  - α is true in all the models in which KB is true



KB |= α **KB |≠ α** **KB |≠ α**

Inference

- Process of verifying the truth of a formula
- Simple inference algorithm:
  - Build a truth table that enumerates all models
  - Verify that formula is true in all models where KB is true

Truth table

P	Q	R	KB
true	true	true	false
true	true	false	false
true	false	true	false
true	false	false	false
false	true	true	false
false	true	false	false
false	false	true	true
false	false	false	true

KB:

$P \Rightarrow Q$,

$Q \Rightarrow R$,

$\neg Q$

$KB \models P$? no

$KB \models \neg P$? yes

$KB \models Q$? no

$KB \models \neg Q$? yes

$KB \models R$? no

$KB \models \neg R$? no

Efficiency

- Building a truth table is inefficient: exponential in the number of variables

- Alternatives: backtracking, resolution
  - Rewrite "KB |= α?" as KB' = (KB and $\neg\alpha$)
  - Show that KB' is **not satisfiable** (e.g., there doesn't exist any model for which KB is true, but α is false)

Convenience

- KB may contain any formula that follows the syntax of our simple logic.
  - Inconvenient: too many possible formula
  - Can we simplify syntax?

- Conjunctive normal form (CNF)
  - Only use $\wedge$, $\vee$, $\neg$
  - Conjunction of clauses, where each clause is a disjunction of (possibly negated) literals
  - Example: $(V1 \vee V2 \vee V3) \wedge (\neg V1 \vee V2) \wedge (\neg V3 \vee V2)$

Logical equivalence

- How do we transform a KB in CNF?
- Logical equivalence rules
  - $\neg(\neg A) \equiv A$
  - $(A \Rightarrow B) \equiv \neg A \vee B$
  - $(A \Leftrightarrow B) \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$
  - $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$
  - $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$
  - $(A \vee (B \wedge C)) \equiv ((A \vee B) \wedge (A \vee C))$
- Can every KB be transformed in CNF in polynomial time and space?
  - No: last rule may yield exponentially many clauses

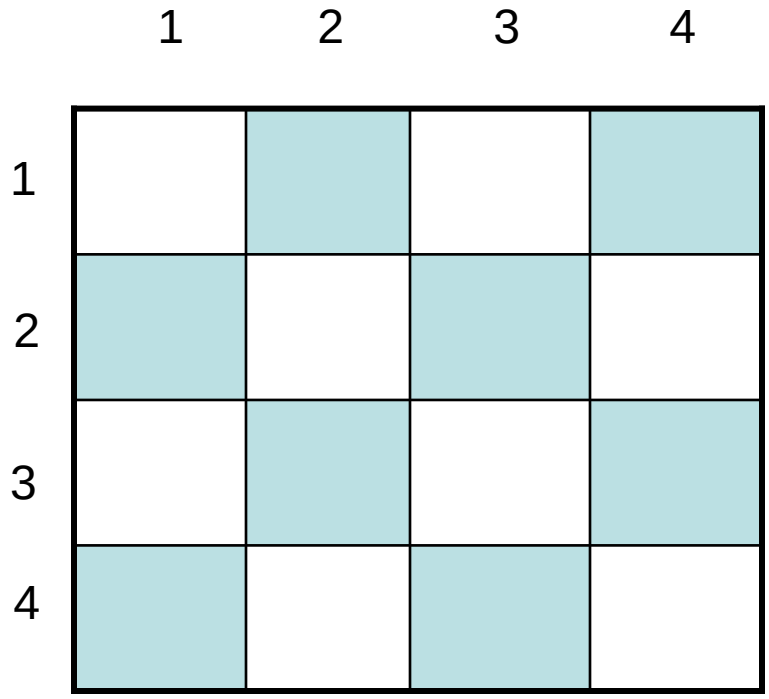
Resolution

- Idea: generate new sentences by implications
  - Unit resolution: $A \wedge (A \Rightarrow B) \models B$
  - General resolution: $(A \Rightarrow B) \wedge (B \Rightarrow C) \models (A \Rightarrow C)$
- How do we do this with KB in CNF?
  - Unit resolution: $A \wedge (\neg A \vee B) \models B$
  - General resolution: $(\neg A \vee B) \wedge (\neg B \vee C) \models (\neg A \vee C)$
- Algorithm: apply resolution to every possible pair of clauses until
  - Empty clause is generated: false KB
  - No more clauses can be generated: all generated clauses are entailed

Concise Language

- Propositional logic provides a general language to encode deterministic domain knowledge.
- While the encoding is precise and well defined it is often tedious or complicated.
- Simple English sentences often lead to long logical formula.
- Can we make propositional logic more concise and more natural?

Recall: 4-Queens problem



- At least one queen in row 1:

$$Q_{11} \vee Q_{12} \vee Q_{13} \vee Q_{14}$$

- At most one queen in row 1:

$$Q_{11} \Rightarrow \neg Q_{12} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{12} \Rightarrow \neg Q_{11} \wedge \neg Q_{13} \wedge \neg Q_{14}$$
$$Q_{13} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{14}$$
$$Q_{14} \Rightarrow \neg Q_{11} \wedge \neg Q_{12} \wedge \neg Q_{13}$$

Repeat for every row, column and diagonal.
This is too tedious! Can we be more concise?

First-order Logic

- World
  - consist of objects and relations
  - Has many objects

- First-order logic:
  - Natural: use predicates to encode relations and constants to encode objects
  - Concise: use quantifiers (e.g., $\forall$, $\exists$) to talk about many objects simultaneously

First-order Logic Syntax

- Sentence → Predicate(Term, …)
  | Term = Term
  | (Sentence Connective Sentence)
  | Quantifier Variable, … Sentence
  | ¬Sentence
- Term → Function(Term, …) | Constant | Variable
- Connective → ⇒ | ∧ | ∨ | ⇔
- Quantifier → ∀ | ∃
- Constant → A | X_1 | John | …
- Variable → a | x | s | …
- Predicate → Before | HasColor | Raining | …
- Function → Mother | LeftLeg | …

Example: kinship domain

- Elizabeth is the mother of Charles
  Mother(Elizabeth,Charles)
- Charles is the father of William
  Father(Charles,William)
- One's grandmother is the mother of one's parent
  $\forall x,z \; \exists y \; Grandmother(x,z) \Leftrightarrow Mother(x,y) \wedge Parent(y,z)$

Symbols

- Constant symbols: objects
  - E.g. William, Elizabeth, Charles
- Predicate symbols: relationships
  - Binary: Mother(,), Grandmother(,)
  - Unary: Female( )
  - Predicates have a truth value
- Function symbols: functions
  - Denote an object
  - E.g.: MotherOf(William) = Elizabeth

Quantifiers

- Universal quantifier: $\forall$
  - For all
  - $\forall x\ P(x) \equiv P(const1) \wedge P(const2) \wedge \ldots$
- Existential quantifier: $\exists$
  - There exists
  - $\exists x\ P(x) \equiv P(const1) \vee P(const2) \vee \ldots$

Nested Quantifiers

- Order of identical quantifiers doesn't matter

- Brothers are siblings
  - $\forall x \; \forall y$ Brother(x,y) $\Rightarrow$ siblings(x,y)
  - $\forall y \; \forall x$ Brother(x,y) $\Rightarrow$ siblings(x,y)

- Similarly for existential quantifiers
  - $\exists x \; \exists y \; P(x,y) \equiv \exists y \; \exists x \; P(x,y)$

Nested Quantifiers

- Order of different quantifiers matters

- $\forall x \exists y$ Loves(x,y)
  - Everyone loves someone
  - Conjunction of disjunctions (CNF)

- $\exists x \forall y$ Loves(x,y)
  - There is a person that loves everyone
  - Disjunction of conjunction (DNF)

Connections between ∀ and ∃

- We only need one of the quantifiers

- De Morgan's rule
  - $\forall x \neg P \equiv \neg \exists x P$ $\quad \neg P \wedge \neg Q \equiv \neg(P \vee Q)$
  - $\neg \forall x P \equiv \exists x \neg P$ $\quad \neg(P \wedge Q) \equiv \neg P \vee \neg Q$
  - $\forall x P \equiv \neg \exists x \neg P$ $\quad P \wedge Q \equiv \neg(\neg P \vee \neg Q)$
  - $\exists x P \equiv \neg \forall x \neg P$ $\quad P \vee Q \equiv \neg(\neg P \wedge \neg Q)$

Equality

- Special relation

- We could define an "Equality" predicate
  - x = y ≡ Equality(x,y)
    - True: when x and y are the same
    - False: otherwise

Practice: 4-Queens problem

- 4-queens problem in first-order logic
- Predicates:
  - Queen(,)
- Constants:
  - 1, 2, 3, 4 (column and row numbers)
- Column constraints:
  - $\forall i, j_1, j_2$ Queen$(i, j_1) \wedge j_1 \neq j_2 \Rightarrow \neg$Queen$(i, j_2)$
  - $\forall i \; \exists j$ Queen(i, j)

Practice: street puzzle

- Predicates:
  - House(), Person( ), Color( ), Drink( ), Job( ), Animal( )
  - Attr(,) (attribute of)
- Constants:
  - 1, 2, 3, 4, 5 (house number)
  - English, Spaniard, Japanese, Italian, Norwegian
  - Red, Green, White, Yellow, Blue
  - Tea, Coffee, Milk, Juice, Water
  - Painter, Sculptor, Diplomat, Violinist, Doctor
  - Dog, Snails, Fox, Horse, Zebra
- Function:
  - Left(x): number of the house to the immediate left of x

Practice: street puzzle

- The Spaniard has a dog
  - $\forall x$ House(x) $\Rightarrow$ (Attr(x,Spaniard) $\Leftrightarrow$ Attr(x,Dog))
- the green house is on the immediate left of the red house
  - $\neg$Attr(1,Red) $\wedge$
  - $\neg$Attr(5,Green) $\wedge$
  - $\forall x$ House(x) $\wedge$ $x \neq 1 \Rightarrow$ (Attr(x,Red) $\Leftrightarrow$ Attr(Left(x),Green))

Propositional vs first-order logic

- Propositional logic:
  - variables
- First-order logic:
  - Quantifiers, predicates, constants, functions

- Syntactically different!
- Are they equally expressive?
  - Prop logic → 1st-order logic: yes
  - 1st-order logic → prop logic: yes (finite domains)
    no (infinite domains)

Propositional → first-order logic

- Variables → predicates
- Indices → constants

- 4-queens problem:
  - $Q_{11} \Rightarrow \neg Q_{12} \wedge \neg Q_{13} \wedge \neg Q_{14}$
  - $Q(1,1) \Rightarrow \neg Q(1,2) \wedge \neg Q(1,3) \wedge \neg Q(1,4)$

First-order → propositional logic

- Quantifiers
  - $\forall$ → conjunction
  - $\exists$ → disjunction
- Predicates → variables
- Constants → indices
- Functions of constants → functions of indices

First-order → propositional logic

- $\forall x\ House(x) \wedge x \neq 1 \Rightarrow$
  $(Attr(x,Red) \Leftrightarrow Attr(Left(x),Green))$

- $\forall x\ House_x \wedge x \neq 1 \Rightarrow (Attr_{x,Red} \Leftrightarrow Attr_{Left(x),Green})$

- $(House_1 \wedge 1 \neq 1 \Rightarrow (Attr_{1,Red} \Leftrightarrow Attr_{Left(1),Green})) \wedge$
  $(House_2 \wedge 2 \neq 1 \Rightarrow (Attr_{2,Red} \Leftrightarrow Attr_{Left(2),Green})) \wedge$
  $(House_3 \wedge 3 \neq 1 \Rightarrow (Attr_{3,Red} \Leftrightarrow Attr_{Left(3),Green})) \wedge$
  $(House_4 \wedge 4 \neq 1 \Rightarrow (Attr_{4,Red} \Leftrightarrow Attr_{Left(4),Green})) \wedge$
  $(House_5 \wedge 5 \neq 1 \Rightarrow (Attr_{5,Red} \Leftrightarrow Attr_{Left(5),Green}))$

Inference

- Propositionalize KB
  - Run favorite ground inference algorithm (e.g., backtracking (DPLL), resolution)
  - Efficient?
    - No: propositionalizing may yield an exponentially large formula
    - $\forall x \exists y \forall z P(x,y,z) \rightarrow$ conjunction of disjunctions of conjunctions
    - $O(n^m)$ where n is # of constants and m is # of quantifiers
- Alternative: lifted inference
  - Work directly with first-order formula

Lifted Inference

- First-order logic:
  - Quantifiers allow us to characterize several objects simultaneously
- Lifted inference:
  - Do inference by reasoning about several terms simultaneously
  - Ground terms only when necessary
  - Hope: determine truth value of goal formula without grounding all terms

Lifted Resolution

Two phases

1. Reduce to lifted conjunctive normal form
2. Resolution with unification

Reduction to Lifted CNF

Six steps

1. Eliminate implications
2. Move negations inwards
3. Standardize variables
4. Skolemize
5. Drop universal quantifiers
6. Distribute $\vee$ over $\wedge$

Reduction to Lifted CNF

Example:
$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

1. Eliminate implications

   $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Move negations inwards (apply DeMorgan's rules)

   $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
   $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
   $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

Reduction to Lifted CNF

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

3. Standardize variables (use different names)

   $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

4. Skolemize (replace existential variables by new constants or functions)

   $\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

Skolemization

- $\exists x\ P(x)$
  - There is at least one term x that satisfies P(x)
  - We don't care what that term is

- Idea: create a new constant
  - Let A be a new constant which could correspond to any object, but since we don't care what that object is, just denote it by this new constant.
  - $\exists x\ P(x) \rightarrow P(A)$

Skolemization

- $\forall x \exists y\ P(x,y)$
  - For each x, there is at least one term y that satisfies P(x,y)
  - We don't care what that term is but it may be different for different x

- Idea: create a new function
  - Let f be a new function that denotes the satisfying term for each x
  - $\forall x \exists y\ P(x,y) \rightarrow P(x,f(x))$

Reduction to Lifted CNF

$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

5. Drop universal quantifiers (since all remaining variables are universally quantified)

   $[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [Loves(G(x),x)]$

6. Distribute $\vee$ over $\wedge$ (to get a CNF)

   $[Animal(F(x)) \vee Loves(G(x),x)] \wedge$
   $[\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Resolution with Unification

- Recall
  - Unit resolution:
    - $A \wedge (A \Rightarrow B) \models B$
    - $A \wedge (\neg A \vee B) \models B$
  - General resolution:
    - $(A \Rightarrow B) \wedge (B \Rightarrow C) \models (A \Rightarrow C)$
    - $(\neg A \vee B) \wedge (\neg B \vee C) \models (\neg A \vee C)$

- Lifted resolution: terms may not match exactly because of variables

Unification

- Variables are like "wild cards" that can be anything

- Find unifier:
  - Unify Knows(John,x) and Knows(John,Jane)
    → Knows(John,Jane) {x/Jane}
  - Unify Knows(John,x) and Knows(y,Bill)
    → Knows(John,Bill) {x/Bill, y/John}
  - Unify Knows(John,x) and Knows(y,Mother(y))
    → Knows(John,Mother(John))
    {x/Mother(John), y/John}

Most General Unifier

- Find unifier that is as general as possible (i.e., preserves as many variables as possible)

- Find most general unifier:
  - Unify Knows(John,x) and Knows(y,Mother(z))
    → Knows(John,Mother(z))
    {y/John, x/Mother(z)}

Resolution with Unification

- Find clauses with positive and negative terms that can be unified
- Eliminate unifying terms and perform substitutions in remaining terms according to most general unifier

- Unit resolution:
  - $A(f(x),x)) \wedge [\neg A(y,g(z)) \vee B(y,z)] \models B(f(g(z)),z)$
  - $\{x/g(z),y/f(g(z))\}$
- General resolution:
  - $[\neg A(x,y) \vee B(y,f(z))] \wedge [\neg B(u,v) \vee C(v,w)] \models [\neg A(x,y) \vee C(f(z),w)]$
  - $\{u/y,v/f(z)\}$

Next class

- Markov Logic Networks