AUTOMATED REASONING- INFERENCE AND KNOWLEDGE PROCESSING

Reasoning in Logic

Contents

- Review of Logic and Propositional Logic Concepts
- Equivalence Laws
- Reasoning
  - Truth Tables method
  - Natural deduction method
  - Axiomatic System
  - Resolution Refutation Method
  - Semantic Tableaux System

Recall: Logic and Propositions

- Logic is a study of principles used to
  - distinguish correct from incorrect reasoning.
- Formally it deals with
  - the notion of truth in an abstract sense and is concerned with the principles of valid inferencing.
- A proposition in logic is a declarative statements which are either true or false (but not both) in a given context. For example,
  - “Nevin is a female”,
  - “Nevin loves Chris" etc.

Recall: Logic and Propositions

- Given some propositions to be true in a given context,
  - logic helps in inferencing new proposition, which is also true in the same context.
- Suppose we are given a set of propositions such as
  - “It is hot today" and
  - “If it is hot it will rain", then
  - we can infer that
    - “It will rain today".
- We can translate
  - simple declarative and
  - conditional (if .. then) natural language sentences into its corresponding propositional formulae.

Recall: Well-formed formula

- Propositional Calculus (PC) is a language of propositions that basically refers
  - to set of rules used to combine the propositions to form compound propositions using logical operators often called connectives such as Λ, V, $\sim$, $\rightarrow$, $\leftrightarrow$
- Well-formed formula is defined as:
  - An atom is a well-formed formula.
  - If α is a well-formed formula, then $\sim\alpha$ is a well-formed formula.
  - If α and β are well formed formulae, then $(\alpha \Lambda \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$ are also well-formed formulae.
  - A propositional expression is a well-formed formula if and only if it can be obtained by using above conditions.

Interpretation and Truth Tables

- Truth table gives us operational definitions of important logical operators.
  - By using truth table, the truth values of well-formed formulae are calculated.
- Truth table elaborates all possible truth values of a formula.
- The meanings of the logical operators are given by the following truth table.

P	Q	~P	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Equivalence Laws

Commutation

1. $P \land Q \cong Q \land P$
2. $P \lor Q \cong Q \lor P$

Association

1. $P \land (Q \land R) \cong (P \land Q) \land R$
2. $P \lor (Q \lor R) \cong (P \lor Q) \lor R$

Double Negation

$\sim(\sim P) \cong P$

Distributive Laws

1. $P \land (Q \lor R) \cong (P \land Q) \lor (P \land R)$
2. $P \lor (Q \land R) \cong (P \lor Q) \land (P \lor R)$

De Morgan's Laws

1. $\sim(P \land Q) \cong \sim P \lor \sim Q$
2. $\sim(P \lor Q) \cong \sim P \land \sim Q$

Law of Excluded Middle

$P \lor \sim P \cong$ T (true)

Law of Contradiction

$P \land \sim P \cong$ F (false)

Key PL Concepts

- PL deals with
  - the validity, satisfiability and unsatisfiability of a formula
  - derivation of a new formula using equivalence laws.
- Each row of a truth table for a given formula is called its **interpretation** under which a formula can be true or false.
- A formula α is called **tautology** if and only
  - if α is true for all interpretations.
- A formula α is also called **valid** if and only if
  - it is a **tautology.**

Key PL Concepts (2)

- Let α be a formula and if there exist at least one interpretation for which α is true,
  - then α is said to be **consistent** (satisfiable) i.e., if $\exists$ a model for α, then α is said to be consistent .
- A formula α is said to be inconsistent (unsatisfiable), if and only if
  - α is always false under all interpretations.

Example

- Show that " It is humid today and if it is humid then it will rain so it will rain today" is a valid argument.
- **Solution:** Let us symbolize English sentences by propositional atoms as follows:

  A : It is humid

  B : It will rain

- Formula corresponding to a text:

  $\alpha : ((A \rightarrow B) \land A) \rightarrow B$

- Using truth table approach, one can see that α is true under all four interpretations and hence is valid argument.

Truth Table for the Example

Truth Table for $((A \rightarrow B) \wedge A) \rightarrow B$				
A	**B**	**$A \rightarrow B = X$**	**$X \wedge A = Y$**	**$Y \rightarrow B$**
T	T	T	T	**T**
T	F	F	F	**T**
F	T	T	F	**T**
F	F	T	F	**T**

Proof and Deduction by Truth Tables (2)

- Truth table method for problem solving is
  - simple and straightforward and
  - very good at presenting a survey of all the truth possibilities in a given situation.
- It is an easy method to evaluate
  - a consistency, inconsistency or validity of a formula, but the size of truth table grows exponentially.
  - Truth table method is good for small values of n.
- For example, if a formula contains n atoms, then the truth table will contain 2^n entries.
  - A formula α : $(P \land Q \land R) \rightarrow (Q \lor S)$ is **valid** can be proved using truth table.
  - A table of 16 rows is constructed and the truth values of α are computed.
  - Since the truth value of α is true under all 16 interpretations, it is valid.

Proof and Deduction by Truth Tables (3)

- We notice that if P Λ Q Λ R is false, then α is true because of the definition of $\rightarrow$.
- Since P Λ Q Λ R is false for 14 entries out of 16, we are left only with two entries to be tested for which α is true.
  - So in order to prove the validity of a formula, all the entries in the truth table may not be relevant.

- P means "It is hot"
- Q means "It is humid"
- R means "It is raining"
- P ^ Q => R

  "If it is hot and humid, then it is raining"
- Q => P

  "If it is humid, then it is hot"
- Q

  "It is humid.“

  Show that(((P ^ Q => R) ^(Q => P)) ^Q)|-R (show that it is raining can be proved)

Other Methods for Proof and Deduction

- Other methods which are concerned with proofs and deductions of logical formula are as follows:
  - Natural Deductive System
  - Resolution Refutation Method
  - Axiomatic System
  - Semantic Tableaux Method

Natural deduction method - ND

- ND is based on the set of few deductive inference rules.
- The name natural deductive system is given because it mimics the pattern of natural reasoning.
- It has about 10 deductive inference rules.

Conventions:

- E for Elimination.
- P, P_k , $(1 \leq k \leq n)$ are atoms.
- α_k, $(1 \leq k \leq n)$ and β are formulae.

Natural Deduction Rules

Rule 1: I-Λ (Introducing Λ) *(And-Introduction)*

I-Λ : If $P_1, P_2, \ldots, P_n$ then $P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n$

Interpretation: If we have hypothesized or proved $P_1, P_2, \ldots$ and P_n, then their conjunction $P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n$ is also proved or derived.

Rule 2: E-Λ (Eliminating Λ) *(And-Elimination)*

E-Λ : If $P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n$ then P_i ($1 \leq i \leq n$)

Interpretation: If we have proved $P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n$, then any P_i is also proved or derived. This rule shows that Λ can be eliminated to yield one of its conjuncts.

Rule 3: I-V (Introducing V) *(Or-Introduction)*

I-V : If P_i ($1 \leq i \leq n$) then $P_1 V P_2 V \ldots V P_n$

Interpretation: If any Pi ($1 \leq i \leq n$) is proved, then $P_1 V \ldots V P_n$ is also proved.

Rule 4: E-V (Eliminating V) *(Or-Elimination)*

E-V : If $P_1 V \ldots V P_n$, $P_1 \rightarrow P, \ldots, P_n \rightarrow P$ then P

Interpretation: If $P_1 V \ldots V P_n$, $P_1 \rightarrow P, \ldots$, and $P_n \rightarrow P$ are proved, then P is proved.

Natural Deduction Rules (2)

Rule 5: I- $\rightarrow$ (Introducing $\rightarrow$)

I- $\rightarrow$: If **from** α_1, …, α_n **infer** β **is proved** then $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \beta$ is **proved**

Interpretation: If given α_1, α_2, …and α_n to be proved and from these we deduce β then $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \rightarrow \beta$ is also proved.

Rule 6: E- $\rightarrow$ (Eliminating $\rightarrow$) - *Modus Ponen*

E- $\rightarrow$: If $P_1 \rightarrow P$, P_1 then P

Rule 7: I- $\leftrightarrow$ (Introducing $\leftrightarrow$)

I- $\leftrightarrow$: If $P_1 \rightarrow P_2$, $P_2 \rightarrow P_1$ then $P_1 \leftrightarrow P_2$

Rule 8: E- $\leftrightarrow$ (Elimination $\leftrightarrow$)

E- $\leftrightarrow$: If $P_1 \leftrightarrow P_2$ then $P_1 \rightarrow P_2$, $P_2 \rightarrow P_1$

Rule 9: I- ~ (Introducing ~)

I- ~ : If **from P infer $P_1 \wedge \sim P_1$** is proved then **~P** is proved

Rule 10: E- ~ (Eliminating ~)

E- ~ : If **from ~ P infer $P_1 \wedge \sim P_1$** is proved then **P** is proved

Natural Deduction ystem

- If a formula β is derived / proved from a set of premises / hypotheses $\{ \alpha_1,\ldots, \alpha_n \}$,
  - then one can write it as **from** $\alpha_1, \ldots, \alpha_n$ **infer** β.
- In natural deductive system,
  - a theorem to be proved should have a form from $\alpha 1, \ldots, \alpha n$ infer β.
- Theorem **infer** β means that
  - there are no premises and β is true under all interpretations i.e., β is a tautology or valid.
- If we assume that $\alpha \rightarrow \beta$ is a premise, then we conclude that β is proved if α is given i.e.,
  - if 'from α infer β' is a theorem then $\alpha \rightarrow \beta$ is concluded.
  - The converse of this is also true.

Deduction Theorem: To prove a formula $\alpha_1 \Lambda \alpha_2 \Lambda \ldots \Lambda \alpha_n \rightarrow \beta$, it is sufficient to prove a theorem **from** $\alpha_1, \alpha_2, \ldots, \alpha_n$ **infer** β.

Examples

Example1: Prove that $P \Lambda (QVR)$ follows from $P \Lambda Q$

Solution: This problem is restated in natural deductive system as "**from P ΛQ infer P Λ (Q V R)**". The formal proof is given as follows:

{Theorem}	**from P ΛQ infer P Λ (Q V R)**	
{ premise}	P Λ Q	(1)
{ E-Λ , (1)}	P	(2)
{ E-Λ , (1)}	Q	(3)
{ I-V , (3) }	Q V R	(4)
{ I-Λ, (2, 4)}	**P Λ (Q V R)**	**Conclusion**

Cont..

Example2: Prove the following theorem:

infer $((Q \rightarrow P) \wedge (Q \rightarrow R)) \rightarrow (Q \rightarrow (P \wedge R))$

Solution:

- In order to prove **infer** $((Q \rightarrow P) \wedge (Q \rightarrow R)) \rightarrow (Q \rightarrow (P \wedge R))$, prove a theorem **from** $\{Q \rightarrow P, Q \rightarrow R\}$ **infer** $Q \rightarrow (P \wedge R)$.
- Further, to prove $\mathbf{Q \rightarrow (P \wedge R)}$, prove a sub theorem **from** Q **infer** $P \wedge R$

{Theorem} from $Q \rightarrow P$, $Q \rightarrow R$ infer $Q \rightarrow (P \wedge R)$

{ premise 1}	$Q \rightarrow P$	(1)
{ premise 2}	$Q \rightarrow R$	(2)
{ sub theorem}	from Q infer $P \wedge R$	(3)
{ premise }	Q	(3.1)
{ E- $\rightarrow$, (1, 3.1) }	P	(3.2)
{E- $\rightarrow$, (2, 3.1) }	R	(3.3)
{ I-$\wedge$, (3.2,3.3) }	$P \wedge R$	(3.4)
{ I- $\rightarrow$, (3)}	$\mathbf{Q \rightarrow (P \wedge R)}$	**Conclusion**

Resolution Refutation in PL

- *Resolution refutation:* Another simple method to prove a formula by contradiction.
- Here negation of goal is added to given set of clauses.
  - If there is a refutation in new set using resolution principle then goal is proved
- During resolution we need to identify two clauses,
  - one with positive atom (P) and other with negative atom (~ P) for the application of resolution rule.
- Resolution is based on modus ponen inference rule.

Disjunctive & Conjunctive Normal Forms

- *Disjunctive Normal Form* (DNF): A formula in the form $(L_{11} \land \ldots\ldots \land L_{1n}) \lor \ldots\ldots \lor (L_{m1} \land \ldots\ldots \land L_{mk})$, where all L_{ij} are literals.
  - Disjunctive Normal Form is **disjunction** of **conjunctions.**
- *Conjunctive Normal Form* (CNF): A formula in the form $(L_{11} \lor \ldots\ldots \lor L_{1n}) \land \ldots\ldots \land (L_{p1} \lor \ldots\ldots \lor L_{pm})$, where all L_{ij} are literals.
  - CNF is **conjunction** of **disjunctions** or
  - CNF is conjunction of clauses
- *Clause:* It is a formula of the form $(L_1 \lor \ldots \lor L_m)$, where each L_k is a positive or negative atom.

Conversion of a Formula to its CNF

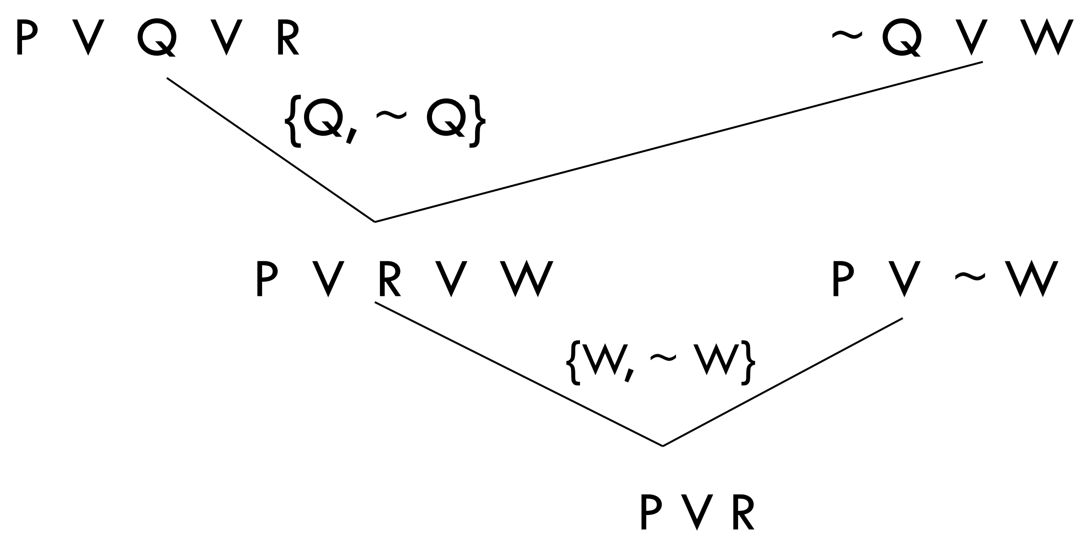
- Each PL formula can be converted into its equivalent CNF.
- Use following equivalence laws:
  - $P \rightarrow Q \cong \sim P \vee Q$
  - $P \leftrightarrow Q \cong (P \rightarrow Q) \wedge (Q \rightarrow P)$
  - Double Negation
    - $\sim \sim P \cong P$
  - (De Morgan's law)
    - $\sim (P \wedge Q) \cong \sim P \vee \sim Q$
    - $\sim (P \vee Q) \cong \sim P \wedge \sim Q$
  - (Distributive law)
    - $P \vee (Q \wedge R) \cong (P \vee Q) \wedge (P \vee R)$

Resolvent of Clauses

- If two clauses C_1 and C_2 contain a complementary pair of literals {L, ~L},
  - then these clauses may be resolved together by deleting L from C_1 and ~ L from C_2 and constructing a new clause by the disjunction of the remaining literals in C_1 and C_2.
- The new clause thus generated is called **resolvent** of C_1 and C_2 .
  - Here C1 and C2 are called parents of resolved clause.
- Inverted binary tree is generated with the last node (root) of the binary tree to be a resolvent.
  - This is also called resolution tree.

Example

- Find resolvent of the following clauses:
  - $C_1 = P \vee Q \vee R$; $C_2 = \sim Q \vee W$; $C_3 = P \vee \sim W$
- Inverted Resolution Tree



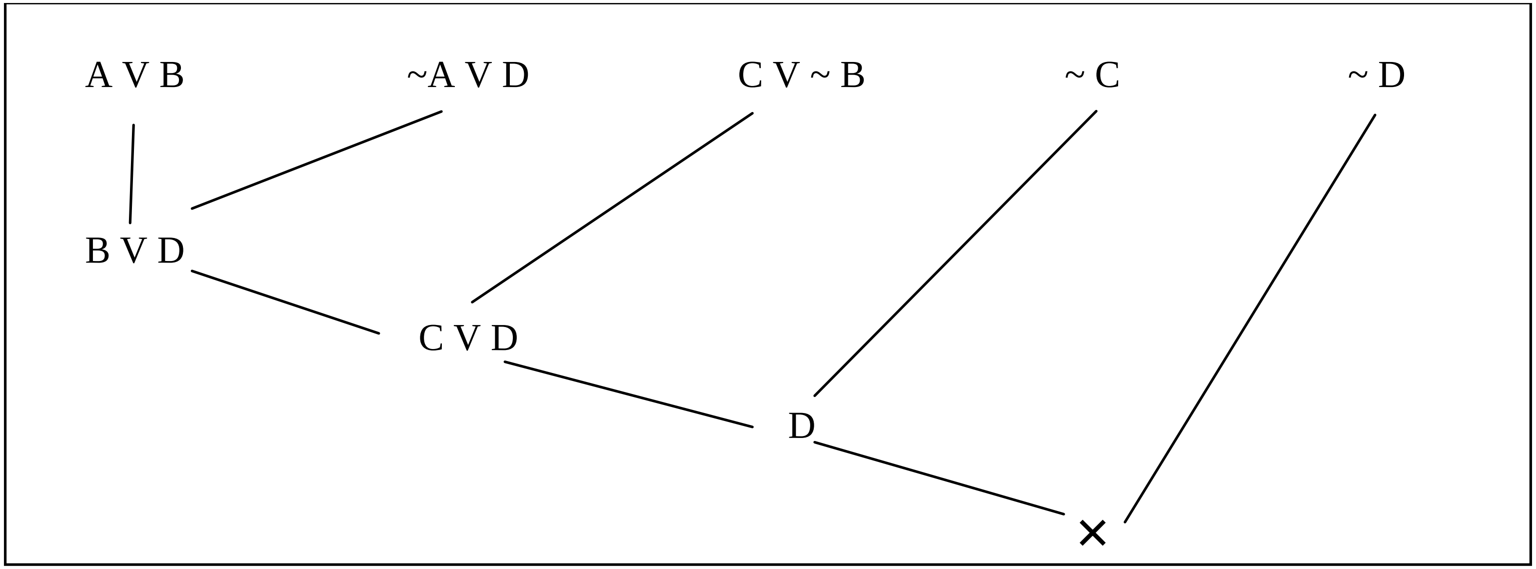
- Resolvent(C1,C2, C3) = P V R

Logical Consequence

- **Theorem1**: If C is a resolvent of two clauses C_1 and C_2, then C is a *logical consequence* of $\{C_1, C_2\}$.
  - A deduction of an empty clause (or resolvent as contradiction) from a set S of clauses is called a *resolution refutation* of S.
- **Theorem2:** Let S be a set of clauses. A clause C is a *logical consequence* of S iff the set $S' = S \cup \{\sim C\}$ is *unsatisfiable*.
  - In other words, C is a logical consequence of a given set S iff an empty clause is deduced from the set S'.

Example

- Show that C V D is a logical consequence of
  - S ={AVB, ~ AVD, C V~ B} using resolution refutation principle.
- First we will add negation of logical consequence
  - i.e., ~ (C V D) ≅ ~C Λ ~D to the set S.
  - Get S’ = {A V B, ~ A V D, C V~ B, ~C, ~D}.
- Now show that S’ is unsatisfiable by deriving contradiction using resolution principle.



Summary

- Review of Propositional Logic (PL) Formula
  - WFF, Interpretation, Tautology, and Validity
- Deduction and Proofs in PL
  - Truth tables
  - Natural deduction
  - Axiomatic systems
  - Resolution refutation
  - Semantic tableaux

Reasoning in FOL

Steps for Resolution Refutation proofs

- Put the premises or axioms into clause form
- Add the negation of what is to be proved, in clause form, to the set of axioms
- Resolve these clauses together, producing new clauses that logically follow from them
- Produce a contradiction by generating the empty clause
- The substitutions used to produce the empty clause are those under which the opposite of the negated goal is true

Putting sentences into clause form

- 1\. Eliminate $\rightarrow$ using
  $a \rightarrow b \equiv \neg a \lor b$
- 2\. Reduce the scope of negations. Transformations include:

  $\neg(\neg a) \equiv a$
- $\neg(\exists X)\, a(X) \equiv (\forall X) \neg a(X)$
- $\neg(\forall X)\, a(X) \equiv (\exists X) \neg a(X)$
- $\neg(a \lor b) \equiv \neg a \land \neg b$
- $\neg(a \land b) \equiv \neg a \lor \neg b$

Putting sentences into clause form (cont'd)

- 3. Standardize variables apart: rename all variables so that variables bound by different quantifiers have unique names
- 4. Move all quantifiers to the left without changing their order
- 5. Eliminate all existential quantifiers using *Skolemization*.
- It's the process of giving a name to an object that must exist.
- 6. Drop all universal quantifiers (allright to do so now)

Putting sentences into clause form (cont'd)

- 7. Convert the expression into a conjunct of disjuncts form
- Eventually each part of an $\wedge$'ed sentence will be separated, and we want the separated sentences to be disjuncts. So,

  $a \wedge (b \vee c)$

  is fine, whereas

  $a \vee (b \wedge c)$ must be distributed to form

  $(a \vee b) \wedge (a \vee c)$

Putting sentences into clause form (cont'd)

- 8. Call each conjunct a separate clause.
- 9. Standardize the variables apart again.

- Using this procedure, any set of statements can be converted to the canonical form.
- Resolution refutation is *complete*, i.e., if a sentence can be entailed (proven) it will be.

More on Skolemization

- It is a simple matter to replace every existentially quantified variable with a unique, new constant and drop the quantifier:

  ∃X (happy (X)) may be replaced by any of the following:

      happy(no-name)
      happy(X#123)
      happy(k1)

  no-name, X#123, and k1 are *Skolem constants*. They should not appear in any other sentence in the KB .

Example

- All people who are graduating are happy.
  All happy people smile.
  John-doe is graduating.
  Goal is to prove:
  Is John-doe smiling?
- First convert to predicate logic
  $\forall X$ graduating(X) $\rightarrow$ happy(X)
  $\forall X$ happy(X) $\rightarrow$ smiling(X)
  graduating (john-doe)

  smiling(john-doe)   negate this: $\neg$ smiling(john-doe)
- Then convert to canonical form

Example (cont'd)

- 1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
  2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
  3. graduating (john-doe)
  4. $\neg$ smiling(john-doe)
- Then convert to canonical form:
- Step 1. Eliminate $\rightarrow$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\neg$ smiling (john-doe)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\neg$ smiling (john-doe)
- Step 2. Reduce the scope of $\neg$
- Step 3. Standardize variables apart
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\neg$ smiling (john-doe)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\neg$ smiling (john-doe)
- Step 4. Move all quantifiers to the left
- Step 5. Eliminate $\exists$
- Step 6. Drop all $\forall$
- 1. $\neg$ graduating (X) $\vee$ happy (X)
  2. $\neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\neg$ smiling (john-doe)

Example (cont'd)

- 1. ¬ graduating (X) ∨ happy (X)
  2. ¬ happy (Y) ∨ smiling (Y)
  3. graduating (john-doe)
  4. ¬ smiling (john-doe)
- Step 7. Convert to conjunct of disjuncts form
- Step 8. Make each conjunct a separate clause.
- Step 9. Standardize variables apart again.
- Ready for resolution!

Example (cont'd)

4. ¬ smiling (john-doe)

2. ¬ happy (Y) ∨ smiling (Y)

{john-doe/Y}

5. ¬ happy (john-doe)

1. ¬ graduating (X) ∨ happy (X)

{john-doe/X}

6. ¬ graduating (john-doe)

3. graduating (john-doe)

7.

Proving an existentially quantified sentence

- All people who are graduating are happy.
  All happy people smile.
  Someone is graduating.
  Goal is prove:
  Is someone smiling?
- First convert to predicate logic
  $\forall X \; graduating(X) \rightarrow happy(X)$
  $\forall X \; happy(X) \rightarrow smiling(X)$
  $\exists \; X \; graduating \; (X)$

  $\exists \; X \; smiling(X)$ negate this: $\neg \exists \; X \; smiling(X)$
- Then convert to canonical form

Example

- 1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
  2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
  3. $\exists$ X graduating (X)
  4. $\neg \exists$ X smiling (X)
- Then convert to canonical form:
- Step 1. Eliminate $\rightarrow$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\exists$ X graduating (X)
  4. $\neg \exists$ X smiling (X)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\exists$ X graduating (X)
  4. $\neg \exists$ X smiling (X)
- Step 2. Reduce the scope of negation.
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\exists$ X graduating (X)
  4. $\forall$ X $\neg$ smiling (X)

Example (cont'd)

- 1. $\forall X \neg$ graduating $(X) \vee$ happy (X)
  2. $\forall X \neg$ happy $(X) \vee$ smiling (X)
  3. $\exists X$ graduating (X)
  4. $\forall X \neg$ smiling (X)
- Step 3. Standardize variables apart
- 1. $\forall X \neg$ graduating $(X) \vee$ happy (X)
  2. $\forall Y \neg$ happy $(Y) \vee$ smiling (Y)
  3. $\exists Z$ graduating (Z)
  4. $\forall W \neg$ smiling (W)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. $\exists$ Z graduating (Z)
  4. $\forall$ W $\neg$ smiling (W)
- Step 4. Move all quantifiers to the left
- Step 5. Eliminate $\exists$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (no-name1)
  4. $\forall$ W $\neg$ smiling (W)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (no-name1)
  4. $\forall$ W $\neg$ smiling (W)
- Step 6. Drop all $\forall$
- 1. $\neg$ graduating (X) $\vee$ happy (X)
  2. $\neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (no-name1)
  4. $\neg$ smiling (W)
- Step 7. Convert to conjunct of disjuncts form
- Step 8. Make each conjunct a separate clause.
- Step 9. Standardize variables apart again.

Example (cont'd)

4. ¬ smiling (W)

2. ¬ happy (Y) ∨ smiling (Y)

{W/Y}

5. ¬ happy (W)

1. ¬ graduating (X) ∨ happy (X)

{W/X}

6. ¬ graduating (W)

{no-name1/W}

3. graduating (no-name1)

7.

Proving a universally quantified sentence

- All people who are graduating are happy.
  All happy people smile.
  Everybody is graduating.
  Goal is to prove:
  Is everybody smiling?
- First convert to predicate logic
  $\forall X\ graduating(X) \rightarrow happy(X)$
  $\forall X\ happy(X) \rightarrow smiling(X)$
  $\forall X\ graduating\ (X)$

  $\forall X\ smiling(X)$ negate this: $\neg \forall\ X\ smiling(X)$
- Then convert to canonical form

Example

- 1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
  2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
  3. $\forall X$ graduating (X)
  4. $\neg \forall$ X smiling (X)
- Then convert to canonical form:
- Step 1. Eliminate $\rightarrow$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\forall X$ graduating (X)
  4. $\neg \forall$ X smiling (X)
-

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\forall X$ graduating (X)
  4. $\neg \forall$ X smiling(X)
- Step 2. Reduce the scope of negation.
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\forall$ X graduating (X)
  4. $\exists$ X $\neg$ smiling (X)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. $\forall$ X graduating (X)
  4. $\exists$ X $\neg$ smiling (X)
- Step 3. Standardize variables apart
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. $\forall$ Z graduating (Z)
  4. $\exists$ W $\neg$ smiling (W)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. $\forall$ Z graduating (Z)
  4. $\exists$ W $\neg$ smiling (W)
- Step 4. Move all quantifiers to the left
- Step 5. Eliminate $\exists$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. $\forall$ Z graduating (Z)
  4. $\neg$ smiling (no-name1)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. $\forall$ Z graduating (Z)
  4. $\neg$ smiling (no-name1)
- Step 6. Drop all $\forall$
- 1. $\neg$ graduating (X) $\vee$ happy (X)
  2. $\neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (Z)
  4. $\neg$ smiling (no-name1)
- Step 7. Convert to conjunct of disjuncts form
- Step 8. Make each conjunct a separate clause.
- Step 9. Standardize variables apart again.

Example (cont'd)

4. ¬ smiling (no-name1)

2. ¬ happy (Y) ∨ smiling (Y)

{no-name/Y}

5. ¬ happy (no-name1)

1. ¬ graduating (X) ∨ happy (X)

{no-name1/X}

6. ¬ graduating (no-name1)

3. graduating (Z)

{no-name1/Z}

7.

Exercise

- All people who are graduating are happy.
  All happy people smile.
- Prove that all people who are graduating smile.

More on Skolemization (cont'd)

- If the existentially quantified variable is in the scope of universally quantified variables, then the existentially quantified variable must be a function of those other variables. We introduce a new, unique function called *Skolem function*.

  $\forall X\ \exists Y$ (loves (X,Y)) may be replaced with
  any of the following:
  $\forall X$ loves (X, no-name(X))
  $\forall X$ loves (X, loved-one(X))
  $\forall X$ loves (X, k1(X))

  no-name, loved-one, k1 are Skolem functions. They should not appear in any other sentence in the KB. They should also not have any other parameter than X.

Resolution refutation algorithm

- Resolution-refutation (KB, α)
- KB $\leftarrow$ KB U { $\neg\, \alpha$ }
- repeat until the null clause is derived
- find two sentences to resolve (should have opposite terms under the mgu)
- KB $\leftarrow$ KB U { the result of resolution }

Example

- All people who are graduating are happy.
  All happy people smile.
  John-doe is graduating.
  Goal:
  Who is smiling?
- First convert to predicate logic
  $\forall X\ graduating(X) \rightarrow happy(X)$
  $\forall X\ happy(X) \rightarrow smiling(X)$
  $graduating\ (john\text{-}doe)$

  $\exists X\ smiling(X)$ negate this: $\neg \exists X\ smiling(X)$
- Then convert to canonical form

Example (cont'd)

- 1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
  2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
  3. graduating (john-doe)
  4. $\neg \exists X$ smiling(X)
- Then convert to canonical form:
- Step 1. Eliminate $\rightarrow$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\neg \exists X$ smiling(X)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\neg \exists X$ smiling(X)
- Step 2. Reduce the scope of $\neg$
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\forall X \neg$ smiling(X)

Example (cont'd)

- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
  3. graduating (john-doe)
  4. $\forall X \neg$ smiling(X)
- Step 3. Standardize variables apart
- 1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
  2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\forall Z \neg$ smiling (Z)

Example (cont'd)

- 1. $\forall X \neg$ graduating $(X) \vee$ happy (X)
  2. $\forall Y \neg$ happy $(Y) \vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\forall Z \neg$ smiling (Z)
- Step 4. Move all quantifiers to the left
- Step 5. Eliminate $\exists$
- Step 6. Drop all $\forall$
- 1. $\neg$ graduating $(X) \vee$ happy (X)
  2. $\neg$ happy $(Y) \vee$ smiling (Y)
  3. graduating (john-doe)
  4. $\neg$ smiling (Z)
- Ready for resolution.

Example (cont'd)

4. ¬ smiling (Z)

2. ¬ happy (Y) ∨ smiling (Y)

{Z/Y}

5. ¬ happy (Z)

1. ¬ graduating (X) ∨ happy (X)

{Z/X}

6. ¬ graduating (Z)

{john-doe/Z}

3. graduating (john-doe)

7.

The substitution for Z is the answer.
John-doe is smiling!