

# Logic Programming (PLP 11)

Predicate Calculus

Clocksin-Mellish Procedure

Horn Clauses

Carlos Varela

Rensselaer Polytechnic Institute

January 28, 2013

# Propositional Logic

- Assigning truth values to logical propositions.
- Formula syntax:

$f$	$::=$	$\vee$	symbol
		$f \wedge f$	and
		$f \vee f$	or
		$f \Leftrightarrow f$	if and only if
		$f \Rightarrow f$	implies
		$\neg f$	not

# Truth Values

- To assign a truth values to a propositional formula, we have to assign truth values to each of its atoms (symbols).
- Formula semantics:

a	b	$a \wedge b$	$a \vee b$	$a \Leftrightarrow b$	$a \Rightarrow b$	$\neg a$
False	False	F	F	T	T	T
False	True	F	T	F	T	T
True	False	F	T	F	F	F
True	True	T	T	T	T	F

# Tautologies

- A *tautology* is a formula, true for all possible assignments.
- For example:  $\neg\neg p \Leftrightarrow p$
- The contrapositive law:

$$(p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p)$$

- De Morgan's law:

$$\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$$

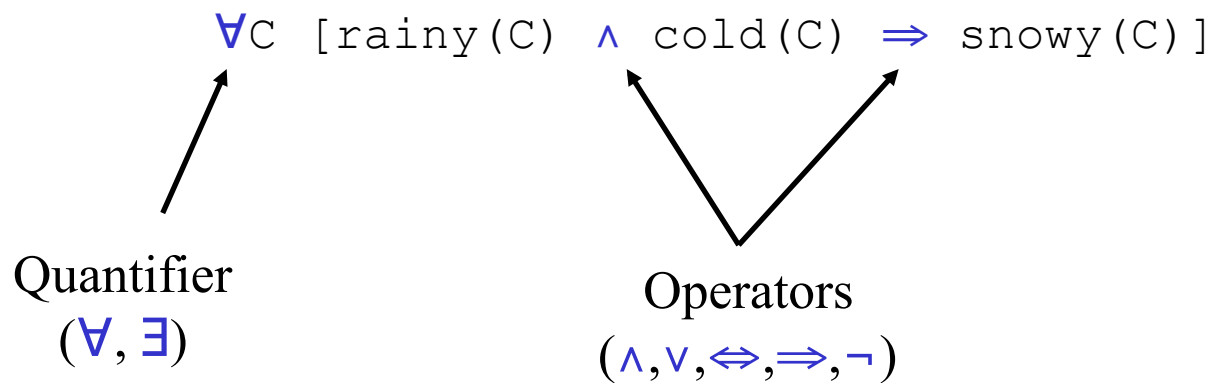
# First Order Predicate Calculus

- Adds variables, terms, and (first-order) quantification of variables.
- Predicate syntax:

a	::=	$p(v_1, v_2, \dots, v_n)$	predicate
f	::=	a	atom
		$v = p(v_1, v_2, \dots, v_n)$	equality
		$v_1 = v_2$	
		$f \wedge f$   $f \vee f$   $f \leftrightarrow f$   $f \Rightarrow f$   $\neg f$	
		$\forall v. f$	universal quantifier
		$\exists v. f$	existential quantifier

# Predicate Calculus

- In mathematical logic, a *predicate* is a function that maps constants or variables to **true** and **false**.
- Predicate calculus enables reasoning about propositions.
- For example:



# Quantifiers

- *Universal* ( $\forall$ ) quantifier indicates that the proposition is true for **all** variable values.
- *Existential* ( $\exists$ ) quantifier indicates that the proposition is true for **at least one** value of the variable.
- For example:

$$\forall A \forall B [\exists C [ \text{takes}(A, C) \wedge \text{takes}(B, C) ] \Rightarrow \text{classmates}(A, B) ]$$

# Structural Congruence Laws

$$P_1 \Rightarrow P_2 \equiv \neg P_1 \vee P_2$$

$$\neg \exists X [P(X)] \equiv \forall X [\neg P(X)]$$

$$\neg \forall X [P(X)] \equiv \exists X [\neg P(X)]$$

$$\neg (P_1 \wedge P_2) \equiv \neg P_1 \vee \neg P_2$$

$$\neg (P_1 \vee P_2) \equiv \neg P_1 \wedge \neg P_2$$

$$\neg \neg P \equiv P$$

$$(P_1 \Leftrightarrow P_2) \equiv (P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$$

$$P_1 \vee (P_2 \wedge P_3) \equiv (P_1 \vee P_2) \wedge (P_1 \vee P_3)$$

$$P_1 \wedge (P_2 \vee P_3) \equiv (P_1 \wedge P_2) \vee (P_1 \wedge P_3)$$

$$P_1 \vee P_2 \equiv P_2 \vee P_1$$



# Clausal Form

- Looking for a *minimal kernel* appropriate for theorem proving.
- Propositions are transformed into *normal form* by using structural congruence relationship.
- One popular normal form candidate is *clausal form*.
- Clocksin and Melish (1994) introduce a 5-step procedure to convert first-order logic propositions into clausal form.

# Clocks in and Melish Procedure

1. Eliminate implication ( $\Rightarrow$ ) and equivalence ( $\Leftrightarrow$ ).
2. Move negation ( $\neg$ ) inwards to individual terms.
3. *Skolemization*: eliminate existential quantifiers ( $\exists$ ).
4. Move universal quantifiers ( $\forall$ ) to top-level and make implicit, i.e., all variables are universally quantified.
5. Use distributive, associative and commutative rules of  $\vee$ ,  $\wedge$ , and  $\neg$ , to move into *conjunctive normal form*, i.e., a conjunction of disjunctions (or *clauses*.)

# Example

$$\forall A [\neg \text{student}(A) \Rightarrow (\neg \text{dorm\_resident}(A) \wedge \neg \exists B [\text{takes}(A,B) \wedge \text{class}(B)])]$$

1. Eliminate implication ( $\Rightarrow$ ) and equivalence ( $\Leftrightarrow$ ).

$$\forall A [\text{student}(A) \vee (\neg \text{dorm\_resident}(A) \wedge \neg \exists B [\text{takes}(A,B) \wedge \text{class}(B)])]$$

2. Move negation ( $\neg$ ) inwards to individual terms.

$$\forall A [\text{student}(A) \vee (\neg \text{dorm\_resident}(A) \wedge \forall B [\neg (\text{takes}(A,B) \wedge \text{class}(B))])]$$

$$\forall A [\text{student}(A) \vee (\neg \text{dorm\_resident}(A) \wedge \forall B [\neg \text{takes}(A,B) \vee \neg \text{class}(B)])]$$

# Example Continued

$$\forall A [\text{student}(A) \vee (\neg \text{dorm\_resident}(A) \wedge \forall B [\neg \text{takes}(A,B) \vee \neg \text{class}(B)])]$$

3. *Skolemization*: eliminate existential quantifiers ( $\exists$ ).
4. Move universal quantifiers ( $\forall$ ) to top-level and make implicit, i.e., all variables are universally quantified.

$$\text{student}(A) \vee (\neg \text{dorm\_resident}(A) \wedge (\neg \text{takes}(A,B) \vee \neg \text{class}(B)))$$

5. Use distributive, associative and commutative rules of  $\vee$ ,  $\wedge$ , and  $\neg$ , to move into *conjunctive normal form*, i.e., a conjunction of disjunctions (or *clauses*.)

$$(\text{student}(A) \vee \neg \text{dorm\_resident}(A)) \wedge (\text{student}(A) \vee \neg \text{takes}(A,B) \vee \neg \text{class}(B))$$

# Clausal Form to Prolog

```
(student(A) ∨ ¬dorm_resident(A)) ∧  
(student(A) ∨ ¬takes(A,B) ∨ ¬class(B))
```

6. Use commutativity of  $\vee$  to move negated terms to the right of each clause.

7. Use  $P_1 \vee \neg P_2 \equiv P_2 \Rightarrow P_1 \equiv P_1 \Leftarrow P_2$

```
(student(A) ⇐ dorm_resident(A)) ∧  
(student(A) ⇐ ¬(¬takes(A,B) ∨ ¬class(B)))
```

8. Move Horn clauses to Prolog:

```
student(A) :- dorm_resident(A).  
student(A) :- takes(A,B), class(B).
```

# Skolemization

$\exists X$  [takes(X, cs101)  $\wedge$  class\_year(X, 2)]

introduce a Skolem constant to get rid of existential quantifier ( $\exists$ ):

takes(x, cs101)  $\wedge$  class\_year(x, 2)

$\forall X$  [ $\neg$ dorm\_resident(X)  $\vee$   
 $\exists A$  [campus\_address\_of(X, A)]]

introduce a Skolem function to get rid of existential quantifier ( $\exists$ ):

$\forall X$  [ $\neg$ dorm\_resident(X)  $\vee$   
campus\_address\_of(X, f(X))]

# Limitations

- If more than one non-negated (positive) term in a clause, then it cannot be moved to a Horn clause (which restricts clauses to only one head term).
- If zero non-negated (positive) terms, the same problem arises (Prolog's inability to prove logical negations).
- For example:
  - « every living thing is an animal or a plant »

`animal(X) ∨ plant(X) ∨ ¬living(X)`

`animal(X) ∨ plant(X) ⇐ living(X)`

# Exercises

4. What is the logical meaning of the second Skolemization example if we do not introduce a Skolem function?
5. Convert the following predicates into Conjunctive Normal Form, and if possible, into Horn clauses:
  - a)  $\forall C [ \text{rainy}(C) \wedge \text{cold}(C) \Rightarrow \text{snowy}(C) ]$
  - b)  $\exists C [ \neg \text{snowy}(C) ]$
  - c)  $\neg \exists C [ \text{snowy}(C) ]$
6. PLP Exercise 11.5 (pg 571).
7. PLP Exercise 11.6 (pg 571).