Natural Deduction for Predicate Calculus

Robert Keller
24 November 2003

Predicate Logic

- Proposition logic does not offer a way to talk about properties of **individuals**.
- This difficulty is overcome in predicate logic, which adds:
  - Constants representing individuals
  - Variables varying over individuals
  - Predicate symbols (including the equality symbol)
  - Function symbols
  - Quantifiers
- We use the same natural deduction framework, just augment the formula language and add new rules.
- “Truth” becomes more complex.

Before: Propositional Language

- E is the start symbol
- E □ A | // Atom
  (□E) | // Negation (not)
  (E □ E) | // Conjunction (and)
  (E   E) | // Disjunction (or)
  (E ‘□ ’ E) | // Implication (implies)
  □ | // Bottom
  T // Top
- A □ ‘p’ | ‘q’ | ‘r’ | ‘s’ | … // Propositions

After: Predicate Language

- E is the start symbol
- E □ A | // Atom (atomic formula)
  (□E) | // Negation (not)
  (E □ E) | // Conjunction (and)
  (E   E) | // Disjunction (or)
  (E ‘□ ’ E) | // Implication (implies)
  □ | // Bottom
  T | // Top
  (□V)E | // Universally-quantified formula
  (□V)E // Existentially-quantified formula
- *A* now requires a more complex production

Atomic Formulas

- A □ P(L) // Predicate applied to list of terms
- L □ T | T ‘,’ L // List of terms
- T □ V | C | F(L) // Term

- V □ ‘x’ | ‘y’ | ‘z’ | … // Variable symbols
- P □ ‘p’ | ‘q’ | ‘r’ | … // Predicate symbols
- C □ ‘a’ | ‘q’ | ‘c’ | … // Constant symbols
- F □ ‘f’ | ‘g’ | ‘h’ | … // Function symbols

Some predicates and functions may be abbreviated in infix form, e.g.
= < < . . . will be infix predicate symbols
\+ * / . . . will be infix function symbols
We will not bother with a special grammar for these, although it can be done.

Examples of Terms

b	constant
y	variable
g(b, y)	function applications
g(h(b), c, h(y))	
g(a, b, g(a, b, c))	

Examples of Atomic Formulas

- p(b)
- q(y)
- p(g(b, y))
- r(a, g(h(b), c, h(y)))

Examples of Quantifier-Free Formulas

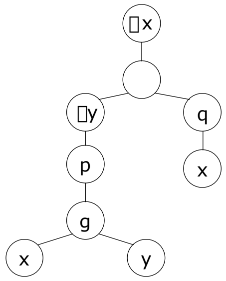
- p(b)   p(c)
- p(y) □ q(y)
- p(g(b, y)) □ q(y)
- □r(a, g(h(b), c, h(y)))

Examples of Formulas

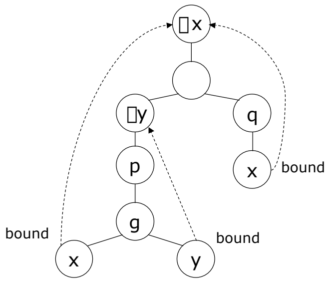
- (□x)p(x)
- (□y) (p(y) □ q(y))
- (□y) (□x) (p(g(x, y)) □ q(y))
- (□x) ((□y) p(g(x, y)))   q(x))

Syntax Trees (or "Parse" Trees)

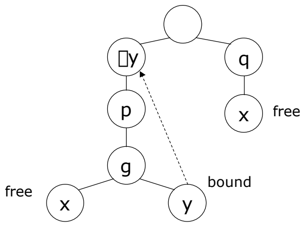
- We are assuming familiarity with syntax trees from CS 60.
- Here (□x) (□x) are treated as 1-ary operators.
- Example: (□x) ((□y) p(g(x, y)))   q(x))



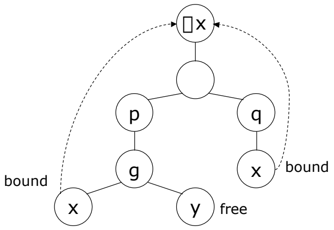
Free and Bound Variable Instances



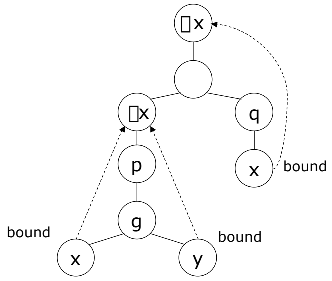
Free and Bound Variable Instances



Free and Bound Variable Instances



Free and Bound Variable Instances



Definition of Free and Bound

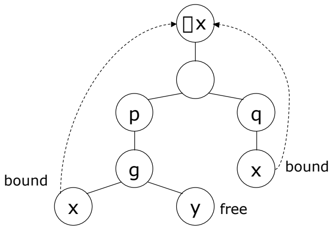
- In a term, every instance of a variable is free.
- If □ is a formula, then any free instances of a variable x become bound in (□x)□ and (□x)□.
- The free instances of variables in □ and □ remain free in (□□), (□ □), (□□□), and (□ □ □).
- The bound instances of variables in □ and □ remain bound in (□□), (□ □), (□□□), and (□ □ □).

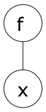
Substitutability

- We are going to need to be able to **substitute** terms for **free** variables in various formulas.
- While this is easy syntactically, there is a semantic restriction that must be observed:
  - In substituting a term for a variable within a formula, **no variables within the term can become bound** as a result of the substitution.
- If t is a term, v is a variable, and F is a formula, and the above restriction applies, we say that

"t is free for v in F."

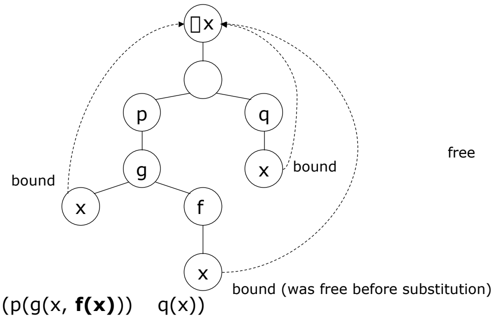
Non-Substitutability Example





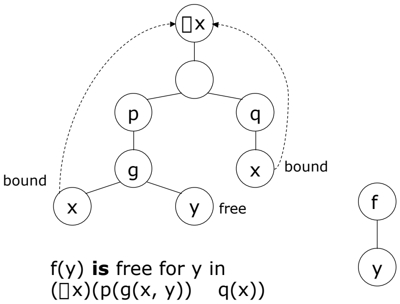
f(x) is **not** free for y in
(□x)(p(g(x, y)) q(x))

Non-Substitutability Example



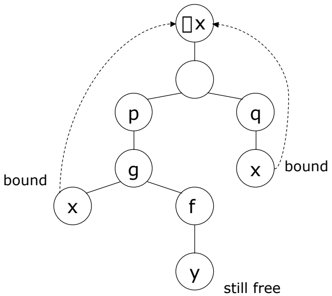
(□x)(p(g(x, **f(x)**)) q(x))

Substitutability Example



f(y) **is** free for y in
(□x)(p(g(x, y))   q(x))

Substitutability Example



(□x)(p(g(x, **f(y)**))   q(x))

Substitution Notation

- If t is a term, v is a variable, and F is a formula, and

  t is free for v in F

  then by

  F[t/v]

  we mean the result of substituting t for every **free** occurrence of v in F.

  This notation is to be used **only** when the substitutability restriction applies.

  Note: [/] is **meta**-syntax; these symbols do not appear in the resulting formula.

Substitution Notation Example

Let F be the formula

(□x)(p(g(x, y))   q(x))

Let v be the variable y.

Let t be the term f(y).

f(y) **is** free for y in (□x)(p(g(x, y))   q(x)).

F[f(y)/y] is (□x)(p(g(x, f(y)))   q(x)).

Substitution Notation Example

Let F be the formula

(□x)(p(g(x, y))   q(x))

Let v be the variable x.

Let t be the term f(y).

f(y) **is** free for x in (□x)(p(g(x, y))   q(x)); there are no free instances of x.

F[f(y)/y] is the same as F.

Natural Deduction Rules

- We need introduction and elimination rules for each of:
  - (□x)
  - (□x)
  - = (as a specially-interpreted predicate symbol)

(□x)-Elimination Rule (□x e)

- (□x) □ (□x e)
  □[t/x]

  where t is any term that is free for x in □.

- **What the rule says:**

  If we have derived a universally-quantified formula □, then the formula □ with any (appropriately-qualified) **specific instance** of x substituted for x is derivable.

Why the Qualification is Necessary

- (□x) □ (□x e)
  □[t/x]

  where t is any term that is free for x in □.

- Correct example: z is free for x in (□y) p(y, x)

1.	(□x) (□y) p(y, x)	Premise	
2.	z = 0	Assumption	
3.	(□y) p(y, z)	(□x e) 1	(substituting **z** for x)

- Incorrect example: y is **not** free for x in (□y) p(y, x)

1.	(□x) (□y) p(y, x)	Premise	
2.	(□y) p(y, y)	(□x e) 1	(substituting **y** for x)

- For instance, p could be > in the domain of natural numbers.

(□x)-Introduction Rule

- This rule uses a sub-derivation, with **no formula assumed**.

x_0
.
.
.
□[x_0/x]
(□x)□ (□x i)

- Here x_0 is a "fresh" variable otherwise unused in the proof.
- x_0 must be free for x in □, but since x_0 is "fresh", this should never be an issue

(□x)-Introduction Rule

- What this rule says:
- If we have argued to derive a term □[x_0/x] where x_0 is an **arbitrary** value of x, then we are justified in concluding (□x)□.
- The key is the word "arbitrary"; there can be no constraints attached to x_0.
- Note: Once the conclusion (□x)□ is drawn, x_0 is **discharged** and cannot be further used.

(□x e) (□x i) Example

- Derive (□x)(p(x) □ q(x)), (□x) p(x) |□ (□x) q(x) :

1.		(□x)(p(x) □ q(x))	Premise
2.		(□x) p(x)	Premise
3.	x_0	p(x_0)	□x e 2
4.		p(x_0) □ q(x_0)	□x e 1
5.		q(x_0)	□ e 3, 4
6.		(□x) p(x)	□x i 3-5

(□x e) (□x i) English Equivalent

- Derive (□x)(p(x) □ q(x)), (□x) p(x) |□ (□x) q(x) :
- Assume (□x)(p(x) □ q(x)) and (□x) p(x).

  Let x_0 be an arbitrary element.

  From the second assumption, we have p(x_0), and the first assumption p(x_0) □ q(x_0), hence also q(x_0) by *modus ponens*.

  Since x_0 was chosen arbitrarily, from q(x_0) we get (□x) q(x).

(□x)-Introduction Rule (□x i)

- □[t/x] (□x i)
  (□x) □

  where t is any term that is free for x in □.

- **What the rule says:**

  If we have exhibited a formula in which □ variable x is replaced by a **specific instance** then we can conclude that there is an x for for which the formula is true.

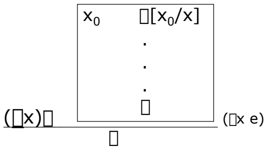
(□x)-Introduction Rule (□x i)

- □[t/x] (□x i)
  (□x) □

  where t is any term that is free for x in □.

- In essence, this rule loses information, by replacing knowledge of a **specific** x for which is true with the statement that there is some such x.
- It is analogous to rule -Introduction.

(□x)-Elimination Rule (□x e)



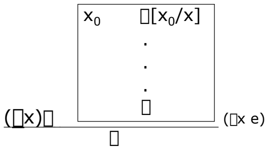
x_0 □[x_0/x]
.
.
.
□

(□x)□ (□x e)

□

- Here x_0 is a “fresh” variable otherwise unused in the proof.
- x_0 must be free for x in □, but since x_0 is “fresh”, this should never be an issue

(□x)-Elimination Rule (□x e)



x_0 □[x_0/x]
.
.
.
□

(□x)□ (□x e)

□

- **What this rule says:**
- Assume that we have derived (□x)□. One use we can make of this fact is to let x_0 be ***an*** x such that □[x_0/x]. There can be no other constraints on x_0. If we then derive □ from the assumption about □, then we can conclude □ in general.

(□x i) (□x e) Example

- Derive (□x)(p(x) □ q(x)), (□x) p(x) |□ (□x) q(x):

1.		(□x)(p(x) □ q(x))	Premise
2.		(□x) p(x)	Premise
3.	x_0	p(x_0)	Assumption
4.		p(x_0) □ q(x_0)	□x e 1
5.		q(x_0)	□ e 3, 4
6.		(□x) q(x)	□x i 5
7.		(□x) q(x)	□x e 3-6

- In the □x e rule, □ is identified with p(x), while □ is identified with (□x) q(x).
- Try not to be confused by the fact that □ is in the conclusion.

(□x i) (□x e) Example in English

- Derive (□x)(p(x) □ q(x)), (□x) p(x) |□ (□x) q(x):
- Assume (□x)(p(x) □ q(x)) and (□x) p(x).

  Let x_0 be such that p(x_0).

  By the first assumption, p(x_0) □ q(x_0).
  Hence q(x_0).

  Since we’ve exhibited an x such that q(x), conclude (□x) q(x).