CSE 321 Discrete Structures

January 13, 2010

Lecture 05

Predicate Calculus and Applications

On the Whiteboard

- Translate from English to predicate calculus (see handout `nested-quantifiers.txt`)

- Renaming quantified variables:

  $\forall x.\ P(x) \equiv \forall y.\ P(y)$

  $\exists x.\ P(x) \equiv \exists y.\ P(y)$

- What is: $\forall x.\ (P(x) \land \exists x. T(x))$?

Natural Deduction

- Existential quantifier:
  - Introduction
  - Elimination
- Universal quantifier:
  - Introduction
  - Elimination
- Plus two "informal" rules
  - Replace equals with equals
  - Rename bound variables whenever needed

Pushing Negations Past Quantifiers

$$\neg \exists x.\ P(x) \equiv \forall x.\ \neg P(x)$$

$$\neg \forall x.\ P(x) \equiv \exists x.\ \neg P(x)$$

$$\neg \exists x.\ \forall y.\ \exists z.\ (\ P(x,y) \vee Q(y,z)\ ) \equiv \ ?$$

Bounded Quantifiers

Suppose we want to restrict x just to D:

$$\exists x. (D(x) \land P(x))$$

$$\forall x. (D(x) \rightarrow P(x))$$

What are these sentences when D is empty ?

Universal Quantifier over Empty Domain

$\forall x. (D(x) \rightarrow P(x))$

What are these sentences when D is empty ?

All flying pigs have titanium tails

True or false ?

Quantifiers over Finite Domains

Suppose the domain has only three elements: a, b, c.

What are the following sentences ?

$\exists x. P(x)$

$\forall x. P(x)$

Quantifiers over Finite Domains

Suppose the domain has only three elements: a, b, c.

What are the following sentences ?

$$\exists x.\ P(x) \equiv P(a) \lor P(b) \lor P(c)$$

$$\forall x.\ P(x) \equiv P(a) \land P(b) \land P(c)$$

Intuitionistic v.s. Classical Proofs

- Intuitionistic proofs requires:

> Whenever you prove $p \vee q$, you must either prove p, or must prove q.

- Similarly:

> Whenever your prove $\exists x. P(x)$ you must find some constant a such that you prove P(a)

- Also known as "constructive proof"

A Nonconstructive Proof

Prove that there exists an irrational number x such that $x^{\sqrt{2}}$ is rational

Let P(x) be the statement
P(x) = "x is irrational and $x^{\sqrt{2}}$ is rational"

- Want to prove $\exists x.\ P(x)$.

Let: $a = \sqrt{2}$, $b = a^{\sqrt{2}}$, $c = b^{\sqrt{2}}$

- Then $c = b^{\sqrt{2}} = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\,\sqrt{2}} = \sqrt{2}^{\,2} = 2$ is rational
- Law of the excluded middle

(b is rational) $\vee$ (b is irrational)

- **Case 1**. If b is rational, then P(a) is true; hence $\exists x.\ P(x)$.
- **Case 2**. If b is irrational, then P(b) is true hence $\exists x.\ P(x)$.

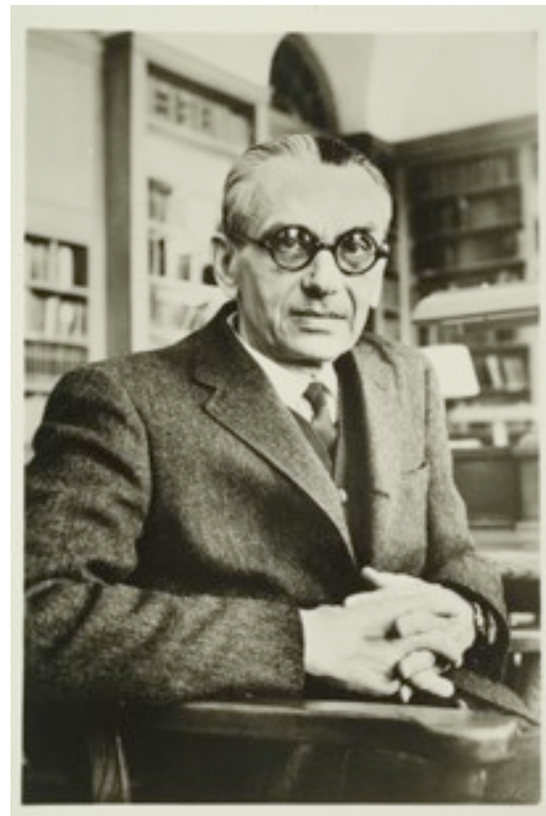
Hence: $\exists x.\ P(x)$

We have proven P(a) $\vee$ P(b), without proving P(a) or P(b)

Proofs and Truth

- What is the connection between proofs and truth ?

Kurt Gödel: 1906-1978



Gödel's completeness theorem

Gödel's incompleteness theorem

Proofs and Truth

- In propositional calculus
  - A **tautology** is a formula that is true for any interpretation of the propositional symbols
- In predicate calculus
  - A **tautology** is a formula that is true for any interpretation of the predicate symbols

- Q: how do we check if P is a tautology (“theorem”) ?
- A: we prove it, $\vdash$ P

Proofs and Truth

- Denote $\vdash$ P if "there exists a proof of P"

SOUNDNESS THEOREM.
If $\vdash$ P, then P is a tautology.

COMPLETENESS THEOREM.
If P is a tautology, then $\vdash$ P

Gödel's completeness theorem

Proofs and Truth

Domain:
Positive Integers

- Now consider ONLY positive integers, and ONLY standard predicates: +, -, *, /, <, >, …
- Suppose a sentence p is true. Can we prove it, $\vdash$ P ?

INCOMPLETENESS. For any proof system that is “reasonable”, there exists a sentence P over positive integers s.t. P is true, and $\nvdash$ P

Natural deduction is “reasonable”

Gödel’s incompleteness theorem

Goldbach's Conjecture

- Every even integer greater than two can be expressed as the sum of two primes

$Prime(x) \equiv \forall y. \forall z. (y*z=x \rightarrow (y=1 \vee y=x))$
$Even(x) \equiv \exists u. x=u+u$

Domain:
Positive Integers

$Goldbach \equiv \forall x. (x > 2 \wedge Even(x)) \rightarrow$
$(\exists y. \exists z. (Prime(y) \wedge Prime(z) \wedge y+z=x))$

Is "Goldbach" a tautology ?

If it is true over positive integers, will we find a proof in Natural Deduction ?

Quantifiers and Nested Loops

Denote $[0..n-1] = \{0,1,\ldots,n-1\}$

Given arrays a[m], b[n], c[p], write programs fragments that check the following properties

$$\forall i \in [0..m-1].\ \forall j \in [0..n-1].\ \exists k.\ \in [0..p-1].\ (a[i]+b[j]=c[k])$$

Quantifiers and Nested Loops

$\forall i \in [0..m-1].\ \forall j \in [0..n-1].\ \exists k. \in [0..p-1].\ (a[i]+b[j]=c[k])$

```
Boolean f = true;
for ( int i = 0; i < m; i++ )
    for ( int j = 0; j < n; j++ )
     { Boolean g = false;
         for ( int k = 0; k < p; k++ )
          if (a[i] + b[j] == c[k]) g = true;
              if (!g) f = false;
          }

if (f) System.out.println("YES");
else System.out.println("NO");
```

Reusing Variables

$$\exists x \exists y \exists z \exists u \exists v.$$
$$((a[x]=b[y]) \wedge (c[y]=d[z]) \wedge (e[z]=f[u]) \wedge (g[u]=h[v]))$$

$$\exists x.(\exists y.(a[x]=b[y] \wedge$$
$$\exists x.(c[y]=d[x] \wedge$$
$$\exists y (e[x]=f[y] \wedge$$
$$\exists x (g[y]=h[x])))))$$

This seems clever. Can we put it to practical use ?

Reusing Variables

$\exists x \exists y \exists z \exists u \exists v.$
$((a[x]=b[y]) \wedge (c[y]=d[z]) \wedge (e[z]=f[u]) \wedge (g[u]=h[v]))$

```
Boolean f = false;
for ( int x = 0; x < n; x++ )
 for ( int y = 0; y < n; y++ )
  for ( int z = 0; z < n; z++ )
   for ( int u = 0; u < n; u++ )
    for ( int v = 0; v < n; v++ )
      if(a[x]==b[y]&&c[y]==d[z]&&e[z]==f[u]&&g[u]==h[v])
          f=true;
```

n^5 iterations

Reusing Variables

$\exists x.(\exists y.(a[x]=b[y] \wedge$
$\exists x.(c[y]=d[x] \wedge$
$\exists y (e[x]=f[y] \wedge$
$\exists x (g[y]=h[x])))))$

$t3[y] = \exists x (g[y]=h[x])$

Reusing Variables

```
Boolean f = false;
for (int x=0; x < n; x++) { t1[x]=f; t2[x]=f; t3[x]=f; }

for ( int x = 0; x < n; x++ )
 for ( int y = 0; y < n; y++ )
      if (g[u]==h[v]) t3[y]=true;

for ( int x = 0; x < n; x++ )
 for ( int y = 0; y < n; y++ )
      if (e[x]==f[v] && t3[y]) t2[x]=true;


for ( int x = 0; x < n; x++ )
 for ( int y = 0; y < n; y++ )
      if (c[y]==d[x] && t2[x]) t1[y]=true;

for ( int x = 0; x < n; x++ )
 for ( int y = 0; y < n; y++ )
       if (a[x]==b[y] && t1[y]) f=true;
```

$4 \times n^2$ iterations