Todays programme: Limitations of Program Verification

- To *prove* fundamental limitations of formalization
  - Program correctness
  - Gödels incompleteness theorem
- To *analyze* and discuss the consequences

Programming language PLN - syntax

- Constants:
  natural numbers: 0, 1, 2,..
  boolean constants: true, false
- *Con* ::= 0, 1, 2, ...
- *Var*::= x, y, z, ...
- *E*::= *Con* | *Var* | *E* + *E* | *E* ∗ *E* | (*E*)
- *B*::= true | false | ¬*B* | *B* ∧ *B* | *B*∨ *B* | *E* = *E* | (*B*)
- *C*::= x := *E* | *C* ; *C* | if *B* then *C* else *C* | while *B* do *C*

PLN specifications syntax

- A correctnes specification of a program *C* is a Hoare triple of the form

  $\{ \phi \}\ C\ \{ \psi \}$

  where ϕ (precondition) and ψ (postcondition) are first order predicate logic formulae over variables (including PLN program variables) and constants/functions/predicates interpreted in the model of natural numbers.

Hoare triples - for *Fac*

```
y := 1; z := 0;
while ¬ (z = x) do
   z := z + 1
   y := y * z
```

- $\models_{par} \{x>5\}\ Fac\ \{z=x\}$
- $\models_{par} \{\ \bar{}\ \}\ Fac\ \{y = x!\}$

Hoare triples - semantics

- $\{ \phi \}\ C\ \{ \psi \}$ is said to be satisfied under partial correctness

  $\models_{par} \{ \phi \}\ C\ \{ \psi \}$

  iff for all states s,

  if $N \models_s \phi$, and $Sem[C](s)$ is defined and equal to s'

  then $N \models_{s'} \psi$

- $\{ \phi \}\ C\ \{ \psi \}$ is said to be satisfied under total correctness

  $\models_{tot} \{ \phi \}\ C\ \{ \psi \}$

  iff for all states s,

  if $N \models_s \phi$, then

  $Sem[C](s)$ **is** defined, and if $Sem[C](s) = s'$ then $N \models_{s'} \psi$

Incompleteness theorem for Hoare triples

- **Theorem**

  There does not exist any sound and complete proof system for *PLN* partial correctness specifications in the form of Hoare triples!

Proof system - definition

- Given a logical language with formulae Φ.
- A *proof system* for Φ consists of an alphabet Σ (for writing proofs) and a set of rules, such that

  for all π in Σ^* and formula Φ,

  it is decidable whether π is a proof of Φ

Proof system - property

- **Theorem**

  For any proof system,

  the set of provable formulae is recursively enumerable

Incompleteness theorem for Hoare triples

- **Theorem**

  There does not exist any sound and complete proof system for *PLN* partial correctness specifications in the form of Hoare triples!

- **Proof**

  SHOW:

  the set of triples $\models_{par} \{\phi\}\ C\ \{\psi\}$ is NOT recursively enumerable!

Post's correspondence problem - example

- List A:

  α_1 = b

  α_2 = babbb

  α_3 = ba

  List B:

  β_1 = bbb

  β_2 = ba

  β_3 = a

  Solution? YES: 2 1 1 3

  $\alpha_2\alpha_1\alpha_1\alpha_3$ = babbb b b ba = babbbbbba

  $\beta_2\beta_1\beta_1\beta_3$ = ba bbb bbb a = babbbbbba

Post's Correspondence Problem *PCP*

- *PCP* instance over alphabet Σ:

  $A = \{w_1, w_2, .., w_k\}$ $B = \{x_1, x_2, .., x_k\}$

  where w_i and x_i are strings over Σ

- *A,B* solution:

  $i_0\ i_1 \ldots\ i_{l-1} \in \{1, 2, .., k\}^+$ such that

  $w_{i0}\ w_{i1}..w_{il-1} = x_{i0}\ x_{i1}..x_{il-1}$

Post's Correspondence Problem *PCP*

- **Theorem**

  The *complement* of *PCP* (i.e. the set of PCP instances with *no* solutions)

  is *not* recursively enumerable

Incompleteness theorem for Hoare triples

Lemma The set of triples $\models_{par} \{\phi\}\ C\ \{\psi\}$ is NOT recursively enumerable!

Proof: Reduction from the complement of *PCP*

Given: An instance of *PCP*, *A,B*

Construct: $\{\phi_{A,B}\}\ C_{A,B}\ \{\psi_{A,B}\}$ such that

A,B has NO solution iff
$\models_{par} \{\phi_{A,B}\}\ C_{A,B}\ \{\psi_{A,B}\}$

Reduction PCP A,B -> $C_{A,B}$ (over strings!)

Given: PCP: $A = \{w_1, w_2, ..w_k\}$ $B = \{x_1, x_2, ...x_k\}$

$C_{A,B}$: let $in = i_0 i_1 .. i_{le-1} \in \{1,2,..,k\}^*$

w := Λ; x := Λ; j:= le;

while j > 0 do

$\{in=i_0..i_{l-1} \wedge w=w_{ij}.. w_{il-1} \wedge x=x_{ij}.. x_{il-1}\}$

j:= j-1;

$w := w_{ij} \bullet w$;

$x := x_{ij} \bullet x$;

if w = x the skip else loop

Representation of strings as numbers

- Given a base number b > 1
- For all $v = i_0 i_1 ... i_{n-1} \in \{0, 1, ..., b-1\}^*$ of length n the b-ary representation of v, $num_b(v)$ is defined as
  $num_b(v) = num_b(i_0 i_1 ... i_{n-1}) = i_0 * b^0 + i_1 * b^1 + ... + i_{n-1} * b^{n-1}$
  $= i_0 + b * num_b(i_1 ... i_{n-1})$
- num_b: $\{0, 1, ..., b-1\}^* \rightarrow N$

Representation of strings as numbers

$num_2(1001) = 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3$ = nine

$num_3(102) = 1*3^0 + 0*3^1 + 2*3^2$ = nineteen

Representation of strings as numbers

- Given a base number $b > 1$
- For any $n \in N$, let

$$rep_b(n) = \begin{cases} \Lambda & \text{if } n = 0 \\ rem(n,b) \bullet rep_b(div(n,b)) & \text{if } n > 0 \end{cases}$$

where $n = rem(n,b) + b*div(n,b)$
and $0 \leq rem(n,b) < b$

- $rep_b: N \rightarrow \{\Lambda\} \cup \{0, 1, ..., b-1\}*\{1, ..., b-1\}$
- $rep_b: N \rightarrow N_b$ (notation)

Representation of strings as numbers

$num_2(1001) = 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3$ = nine

$num_3(102) = 1*3^0 + 0*3^1 + 2*3^2$ = nineteen

$rep_2(six) = 0 \bullet rep_2(three) = 0 \bullet (1 \bullet rep_2(one)) = 0 \bullet (1 \bullet 1)$
$= 011$

$rep_3(eleven) = 2 \bullet rep_3(three) = 2 \bullet (0 \bullet rep_3(one))$
$= 2 \bullet (0 \bullet 1) = 201$

Representation of strings as numbers

- **Propositions**

For all $b>1$

For all $n \in N$,
$num_b(rep_b(n)) = n$

For all $w \in N_b$,
$rep_b(num_b(w)) = w$

i.e. num_b and rep_b are bijections between N and N_b!

Representation of strings as numbers

- **Propositions**

For all $n, i \in N, 0 \leq i < |rep_b(n)|$
the (unique) i'th digit in $rep_b(n)$ is: $rem(div(n, b^i), b)$

For all $v, w \in N_b$
$num_b(vw) = num_b(v) + num_b(w) * b^{|v|}$

- Question

Can the operations above be computed in *PLN*?

PLN macros

- ”x := monus (m, n)”
  where monus (m, n) = m - n, if m > n
  0, otherwise

  can be computed in PLN by:

  x:= 0; y:= 0;
  while ¬(y = m ∨ y = n) do y := y+1;
  while ¬(y = m) do
  y := y+1; x := x+1

PLN macros

- ”m ≤ n”
  can be computed in PLN by:
  monus (m, n) = 0

- ”m > n”
  can be computed in PLN by:
  ¬(monus (m, n) = 0)

PLN macros

- ”d := div(m, n)” (integer division of m by n, where n ≠ 0)
- ”r := rem(m,n)” (remainder of int.div. of m by n, n≠ 0)
- (m = d*n + r, where 0 ≤ r < n)

  can be computed in PLN by

  d:= 0;
  while ((d + 1)*n ≤ m) do d := d + 1;
  r:= monus (m, d *n)

PLN macros

- ”x := m ↑ n” (”m to the power n”)

  can be computed in PLN by

  x := 1; y := 0;
  while ¬(y = n) do
  x := x*m; y := y+1

Incompleteness theorem for Hoare triples

Given: PCP over Σ: A = $\{w_1, w_2,..w_k\}$ B = $\{x_1, x_2,...x_k\}$
where Σ = {1, 2,.., |Σ| }

Construct: $\phi_{A,B} = \top$ $\psi_{A,B} = \bot$
and $C_{A,B}$ such that

PCP has NO solution iff
$\models_{par} \{\top\}\ C_{A,B}\ \{\bot\}$
(i.e. iff $C_{A,B}$ diverges for all initial states)

Reduction PCP A,B -> $C_{A,B}$ (over strings!)

Given: PCP: A = $\{w_1, w_2,..w_k\}$ B = $\{x_1, x_2,...x_k\}$

$C_{A,B}$: let in = $i_0 i_1..i_{le-1} \in \{1,2,..,k\}^*$
w := Λ; x := Λ; j:= le;
while j > 0 do
{in=$i_0..i_{l-1}$ ∧ w=$w_{ij}..\ w_{il-1}$ ∧ x=$x_{ij}..\ x_{il-1}$}
j:= j-1;
w := w_{ij} • w;
x := x_{ij} • x;
if w = x the skip else loop

Reduction PCP A,B -> $C_{A,B}$ - intuition

- Given a number *in* (input)
  - Convert *in* to a string of small numbers $rep_b(in)$
  - View this string as a potential solution to PCP A,B
  - Construct (the num_b-versions of) the corresponding concatenation of A- and B-strings
  - Check for equality of these numbers
    - If equal: terminate, if not: loop!
- Claim: this algorithm terminates for some input iff A,B has a solution!

Reduction PCP A,B -> $C_{A,B}$

- Given PCP $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$ over Σ = {1, 2,.., |Σ| } !!!!
- Compute base number b = max{k, |Σ| } +1 and constants $num_b(w_i)$ and $num_b(x_i)$, $|w_i|$ and $|x_i|$

PLN shorthand notation

- ”skip”
  shorthand for ”y := y”
- ”loop”
  shorthand for ”while true do skip”

Reduction PCP A,B -> $C_{A,B}$

Given: PCP: A = {w_1, w_2,..w_k} B = {x_1, x_2,...x_k}

$C_{A,B}$: if in=0 then loop; j:= 1; while div(in,b^j) > 0 do j:= j+1;
w := 0; x := 0;
while j > 0 do
{$rep_b(in)=i_0..i_{le-1} \wedge w=num_b(w_{ij}.. w_{ile-1}) \wedge x=num_b(x_{ij}.. x_{ile-1})$}
j:= j-1; i := rem(div(in, b^j), b);
if i = 1 then w := $num_b(w_1)$ + w*($b\uparrow|w_1|$);
x := $num_b(x_1)$ + x*($b\uparrow|x_1|$) else
.....
if i = k then w := $num_b(w_k)$ + w*($b\uparrow|w_k|$);
x := $num_b(x_k)$ + x*($b\uparrow|x_k|$) else loop
if w = x the skip else loop

Reduction PCP A,B -> $C_{A,B}$

Example: PCP: A = {12, 2} B = {1, 22}
$C_{A,B}$: ??

Reduction PCP A,B -> $C_{A,B}$

Example: PCP: A = {12, 2} B = {1, 22}
$C_{A,B}$: if in=0 then loop; j:= 1; while div(in,b^j) > 0 do j:= j+1;
w := 0; x := 0;
while j > 0 do
j:= j-1; i := rem(div(in, 3^j), 3);
if i = 1 then w := 7 + w*($3\uparrow 2$);
x := 1 + x*($3\uparrow 1$) else
if i = 2 then w := 2 + w*($3\uparrow 1$);
x := 8 + x*($3\uparrow 2$) else loop
if w = x the skip else loop

Incompleteness theorem for Hoare triples

- Claim:

  A,B has solution iff

  $C_{A,B}$ terminates for some input

Todays programme: Limitations of Program Verification

- To *prove* fundamental limitations of formalization
  - Program correctness
  - Gödels incompleteness theorem
- To *analyze* and discuss the consequences

Hoare proof rules if and while

$$\frac{\{\phi \wedge B\}\ C_1\ \{\psi\} \qquad \{\phi \wedge \neg B\}\ C_2\ \{\psi\}}{\{\phi\}\ \text{if } B \text{ then } C_1 \text{ else } C_2\ \{\psi\}}\ \text{If-rule}$$

$$\frac{\{\psi \wedge B\}\ C\ \{\psi\}}{\{\psi\}\ \text{while } B\ \text{do } C\ \{\psi \wedge \neg B\}}\ \text{While-rule}$$

A proof of Euclid's gcd algorithm

$\{ m = m_0 \geq 1 \wedge n = n_0 \geq 1 \}$
while $\neg$ (m = n) do
$\{ gcd(m,n) = gcd(m_0, n_0) \}$
if m > n then m:=m-n
else n:= n-m;
$\{m = gcd(m_0, n_0) \}$
r:= m
$\{ r = gcd(m_0, n_0) \}$

$\{gcd(m,n) = gcd(m_0, n_0) \wedge \neg (m = n) \}$
if m > n then m:=m-n
else n:= n-m;
$\{gcd(m,n) = gcd(m_0, n_0) \}$

$\{gcd(m,n) = gcd(m_0, n_0) \}$
while ...
$\{gcd(m,n) = gcd(m_0, n_0) \wedge \neg \neg(m = n) \}$
While-rule

A proof of Euclid's gcd algorithm

$\{ m = m_0 \geq 1 \wedge n = n_0 \geq 1 \}$

while $\neg$ (m = n) do

$\{ gcd(m,n) = gcd(m_0, n_0) \}$

if m > n then m:=m-n

else n:= n-m;

$\{m = gcd(m_0, n_0) \}$

r:= m

$\{ r = gcd(m_0, n_0) \}$

Proof obligations Comp rule:

$\vdash_N m = m_0 \geq 1 \wedge n = n_0 \geq 1$

$\rightarrow$ $gcd(m,n) = gcd(m_0,n_0)$

$\vdash_N gcd(m,n)= gcd(m_0,n_0) \wedge\neg\neg(m=n)$

$\rightarrow$ $m = gcd(m_0, n_0)$

Hoare proof rules - implied

$$\frac{\vdash_N \phi' \rightarrow \phi \qquad \{\phi\}\ C\ \{\psi\} \qquad \vdash_N \psi \rightarrow \psi'}{\{\phi'\}\ C\ \{\psi'\}}$$ Impl-rule

NOTE We assume here that we have some underlying extension of the proof system for predicate logic, in which we prove formulae of the form $\phi' \rightarrow \phi$ valid for N - the model of natural numbers!!!!

A simple *N* vocabulary

- Let $N_{\uparrow}$ be the predicate logic interpretation with
  - natural numbers as the univers of values,
  - constants 0,1
  - function symbols +, ×, and ↑
  - predicate symbol =

  all interpreted "as usual"

Peano proof rules

$$\frac{}{\forall n.\ \neg(n = n+1)} \qquad \frac{}{\forall m\ \forall n.\ (m+1 = n+1) \rightarrow (m = n)}$$

$$\frac{}{\forall n.\ n+0 = n} \qquad \frac{}{\forall m\ \forall n.\ m+(n+1) = (m+n) + 1}$$

$$\frac{}{\forall n.\ n*0 = 0} \qquad \frac{}{\forall m\ \forall n.\ m*(n+1) = (m*n) + m}$$

$$\frac{}{\forall n.\ n\uparrow 0 = 1} \qquad \frac{}{\forall m\ \forall n.\ m\uparrow(n+1) = (m\uparrow n) * m}$$

$$\frac{\varphi(0) \qquad \forall n.\ (\varphi(n) \rightarrow \varphi(n+1))}{\forall n.\ \varphi(n)}$$

Incompleteness of Peano axioms

$\forall\, \forall n.\ (\quad (\neg\, (n = 0)\ ) \rightarrow\ \exists\, m\ (n = m+1)\quad)$

can NOT be shown in Peano's proof system

Gödel's incompleteness theorem

- There does not exist any sound and complete proof system for the model of natural numbers $N_{\uparrow}$ with constant 0,1, function symbols +, $*$, $\uparrow$, and predicate symbol =
- Proof

  SHOW: The set $\{\phi \mid N_{\uparrow} \models \phi\}$

  is NOT recursively enumerable!

Gödel's incompleteness theorem

Lemma: The set $\{\phi \mid N_{\uparrow} \models \phi\}$ is NOT recursively enumerable!

Proof: reduction from the complement of PCP

Given: An instance of PCP: *A,B*

Construct: $\phi_{A,B}$ such that

A,B has NO solution iff

$N_{\uparrow} \models \phi_{A,B}$

Gödel's incompleteness theorem

Given: PCP over Σ: $A = \{w_1, w_2, ..w_k\}$ $B = \{x_1, x_2, ...x_k\}$

compute

$b > \max\{k, |\Sigma|\} + 1$

$\max = \max\{|w_i|, |x_i| \mid 1 \le i \le k\}$

and for $1 \le i \le k$:

$num_b(w_i)$

$|w_i|$

$num_b(x_i)$

$|x_i|$

Reduction PCP A,B -> $\psi_{A,B}$

Given: PCP: $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$

$\psi_{A,B}$: $\exists$ le. $\exists$ m. (le $\geq$ l) $\wedge$ (m $\geq$ le$*$max) $\wedge$
$\exists$ w, x. FIRST(m,w,x) $\wedge$ NEXT (le,m,w,x) $\wedge$ LAST (le,m,w,x)

$N_{\uparrow}$ expressiveness

- "m $\geq$ n"
  may be expressed as "$\exists$ a. (m = n + a)"
- "m > n"
  may be expressed as "(m $\geq$ n $\wedge$ $\neg$(m = n))"

Operations on strings in arithmetic

- Assume n = $num_b(i_0 i_1 ... i_j .....)$

$div(n, b^j) = num_b(i_j i_{j+1,} .....)$
$rem(n, b^j) = num_b(i_0 i_1, ... i_{j-1})$

$sel(n, j, k) = rem(\ div(n, b^j), b^k) = num_b(i_j, i_{j+1} ... i_{j+k-1})$

$N_{\uparrow}$ expressiveness

- "div(m, n) = d" (d is integer division of m by n)
  may be expressed as "$\exists$ r. (m = n$*$ d + r $\wedge$ r < n)"
- "rem(m, n) = r" (r is remainder of int. division of m by n)
  may be expressed as "m = n$*$ div(m, n) + r "
- "sel_b(m, j, k)" (the number repesented by the k digits starting from digit j in rep_b(m)0$*$
  may be expressed as
  " $sel_b(m, j, k) = rem(\ div(m, b\uparrow j),\ b\uparrow k)$)"

Reduction PCP A,B -> $\psi_{A,B}$

Given: PCP: $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$

$\psi_{A,B}$: $\exists$ le. $\exists$ m. (le $\geq$ l) $\wedge$ (m $\geq$ le*max) $\wedge$
$\exists$ w, x. FIRST(m,w,x) $\wedge$ NEXT (le,m,w,x) $\wedge$ LAST (le,m,w,x)

FIRST(m,w,x) : $sel_b(w, 0, m) = 0 \quad \wedge \quad sel_b(x, 0, m) = 0$

Reduction PCP A,B -> $\psi_{A,B}$

Given: PCP: $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$

$\psi_{A,B}$: $\exists$ le. $\exists$ m. (le $\geq$ l) $\wedge$ (m $\geq$ le*max) $\wedge$
$\exists$ w, x. FIRST(m,w,x) $\wedge$ NEXT (le,m,w,x) $\wedge$ LAST (le,m,w,x)

FIRST(m,w,x) : $sel_b(w, 0, m) = 0 \quad \wedge \quad sel_b(x, 0, m) = 0$

LAST(le,m,w,x): $sel_b(w, m*le, m) = sel_b(x, m*le, m)$

Reduction PCP A,B -> $\psi_{A,B}$

Given: PCP: $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$

NEXT: $\forall j. ((0 \leq j \wedge j<le) \rightarrow$
$\exists i. (1 \leq i \wedge i \leq k \wedge$ MATCH(m,w,x,j,i)

MATCH(m,w,x,j,i):
$(i=1 \rightarrow sel_b(w, (j+1)*m, m) = num_b(w_1) + sel_b(w, j*m, m) *b^{|w1|}$
$\wedge sel_b(x, (j+1)*m, m) = num_b(x_1) + sel_b(x, j*m, m) *b^{|x1|})$
$\wedge$ .....
$(i=k \rightarrow sel_b(w, (j+1)*m, m) = num_b(w_k) + sel_b(w, j*m, m) *b^{|wk|}$
$\wedge sel_b(x, (j+1)*m, m) = num_b(x_k) + sel_b(x, j*m, m)*b^{|xk|})$

Reduction PCP A,B -> $\psi_{A,B}$

Given: PCP: $A = \{w_1, w_2,..w_k\}$ $B = \{x_1, x_2,...x_k\}$

CLAIM: *A,B* has NO solution iff $N_\uparrow \models \phi_{A,B}$ $(= \neg\psi_{A,B})$

i.e

A,B HAS a solution iff $N_\uparrow \models \psi_{A,B}$

Todays programme: Limitations of Program Verification

- To *prove* fundamental limitations of formalization
  - Program correctness
  - Gödels incompleteness theorem
- To *analyze* and discuss the consequences

Exercises

- All exercises in the following referring to the note Limitations of Program Verification [LiProVer07]
- *Describe* representations of numbers
  - [LiProVer07] 2 (p 16), 3, 4 (p 17) : understanding number representations
  - [LiProVer07] 9 (p 23): understanding selection predicate
- Prove limitations of formalization
  - [LiProVer07] 5, 6, 8 (p 21): understanding the reduction to Hoare specifications
  - [LiProVer07] 10 (p 27): understanding the reduction to predicate logic over the natural numbers

dBerLog Compulsory Assignments 2007

- Write manuscripts for a 15 minutes exam presentation for each of the two exam questions: Computability and Logic
- 2-3 pages each
- dBerLog curriculum follows from dBerLog home page - Weekly Schedules (and Final Exam)
- Second assignment: Logic
- Hand in to your tutor no later than Wednesday October 10! OBS: HARD DEADLINE!!!!!