Todays programme:
Predicate Logic and Program Verification

- *Familiarity* with basic *concepts/results* of predicate logic
  - Syntax: variables, quantification, scope
  - Semantics: interpretations, valuations, satisfaction truth, validity
  - Axiomatic proof system FOPL
  - Gödels completeness theorem for predicate logic
- *Describe* the *use* of predicate logic in program verification
  - Syntax: program specifications, Hoare triples
  - Semantics: partial and total correctness
  - Proof system: Hoare proof rules

Predicate Logic

- Sten kan ikke flyve og morlille kan ikke flyve
  ergo er morlille en sten!
- $(\forall x. (S(x) \rightarrow \neg F(x))) \wedge \neg F(morlille)) \quad |\!=\!/ \; S(morlille)$

- Fugle kan flyve og piphans er en fugl
  ergo kan piphans flyve!
- $(\forall x. (B(x) \rightarrow F(x))) \wedge B(piphans)) \quad |\!= \quad F(piphans)$

Predicate Logic

Female(girl).
Floats(duck).
Sameweigth(girl, duck).
Witch(X) :- Burns(X).
Burns(X) :- Wooden(X).
Wooden(X) :- Floats(X).
Floats(X) :- Sameweight(X, Y), Floats(Y).

Witch(girl)?

Predicate Logic

Female(girl),
Floats(duck),
Sameweigth(girl, duck),
$\forall$x Witch(x) $\leftarrow$ Burns(x),
$\forall$x Burns(x) $\leftarrow$ Wooden(x),
$\forall$x Wooden(x) $\leftarrow$ Floats(x),
$\forall$ x,y (Floats(x) $\leftarrow$ Sameweight(x, y) $\wedge$ Floats(y))
|= ?
Witch(girl)

Predicate Logic - syntax examples

- Constants: girl, duck
- Predicate symbols **P**: Female, Floats,.... with arity 1
  Sameweight with arity 2

Predicate Logic for Natural Numbers

- $\forall x.\ Even(x) \rightarrow Even(succ(succ(x)))$
- $\forall x.\ \forall y.\ (Even(x) \wedge y = x+2) \rightarrow Even(y)$
- $\forall x.\ x + 0 = x$
- $(A(0) \wedge (\forall x.\ A(x) \rightarrow A(x+1)) \rightarrow \forall x.\ A(x)$

Predicate Logic - syntax examples

- Constants: girl, duck
- Predicate symbols **P**: Female, Floats,.... with arity 1
  Sameweight with arity 2

- Constants 0,1,2,...
- Function symbols **F**: $+, \times$ both with arity 2
- Predicate symbols **P**: $=$ with arity 2

Predicate Logic - syntax

- Variables x,y,z,...
- Constants **C**: c_1, c_2,....
- Function symbols **F**: f,g,h... each with some arity n>0

- Terms
  $t ::= c \mid x \mid f(t_1, t_2,..t_n)$

Predicate Logic - first order language, wwf's

- Predicate symbols **P**: P, Q, R each with some arity $n \geq 0$
- Well formed formulae *wff*:

Φ ::= $P(t_1, t_2, .., t_n)$ |

$\neg \Phi$ | $\Phi \vee \Phi$ | $\Phi \wedge \Phi$ | $\Phi \rightarrow \Phi$ |

$\forall x \Phi$ | $\exists x \Phi$

Predicate Logic - Interpretations

- An interpretation **I** for a first order predicate logic language consists of

D, a domain of concrete values

for each constant $c^{\mathbf{I}}$ an element of D

for each $f \in \mathbf{F}$ with arity n, a function $f^{\mathbf{I}}: D^n \rightarrow D$

for each $P \in \mathbf{P}$ with arity n, a subset $P^{\mathbf{I}} \subseteq D^n$

Predicate Logic - interpretations example

- D: objects from the real world

girl: the girl in question

duck: the duck on the scales

Female: those objects which are female

Sameweight: those pairs of objects with the same weight

I $\models$ ¬Wooden(girl) ∧ ¬Witch(duck)

I $\models$ ∃ x Female(x) since **I** $\models$ Female(girl)

Predicate Logic

Female(girl),
Floats(duck),
Sameweigth(girl, duck),
∀x Witch(x) ← Burns(x),
∀x Burns(x) ← Wooden(x),
∀x Wooden(x) ← Floats(x),
∀ x,y (Floats(x) ← Sameweight(x, y) ∧ Floats(y))
|= ?
Witch(girl)

Predicate Logic - interpretations example

- D: Natural numbers, N

  0,1,...: the numbers zero, one,...

  $+, \times$: sum and mutiplication on N

  =: equality on N

I $\models \forall$x. x + 0 = x

I $\models$ $\forall$ x $\exists$ y (y = x+1)

I $\models$ x + 1 = y?

Predicate Logic - valuations

- A valuation *v* in an interpretation **I** of a first order language is a function from the terms of L to the domain D of **I** such that

  $v(c) = c^{I}$ for all constants

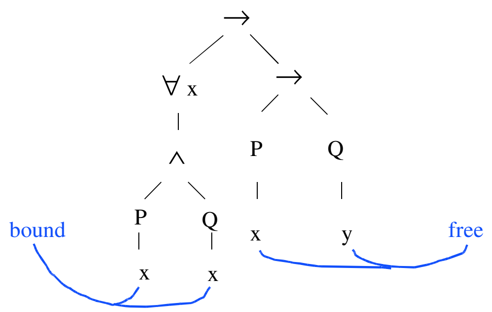
  $v(x) \in$ D for all variables x

  for each f $\in$ **F** with arity n, $v(f(t_1,..,t_n)) = f^{I}(v(t_1),..,v(t_n))$
- That is essentially a ”look-up table”

  v: *free* variables $\rightarrow$ D

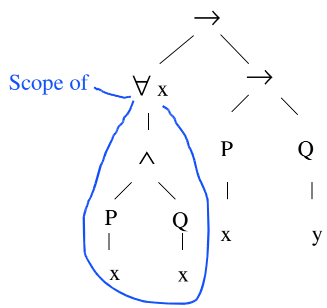
Predicate logic - free and bound variables

- $(\forall x\ (P(x) \wedge Q(x)) \rightarrow (P(x) \rightarrow Q(y))$



Predicate logic - free and bound variables

- $(\forall x\ (P(x) \wedge Q(x)) \rightarrow (P(x) \rightarrow Q(y))$



Predicate Logic - satisfaction (semantics)

- Given an interpretation, **I**, for a first order language, a valuation v, and a formula A, v *satisfies* A

- $\mathbf{I} \models_v A$ iff

  if $A = P(t_1, t_2,.., t_n)$ then $(v(t_1), v(t_2),.., v(t_n)) \in P^{\mathbf{I}}$

  if $A = \forall x\, B$ then $\mathbf{I} \models_{v[x \leftarrow d]} B$ for all $d \in D$

  if $A = \exists x\, B$ then $\mathbf{I} \models_{v[x \leftarrow d]} B$ for some $d \in D$

  if $A = \neg B, B \vee C, B \wedge C, B \rightarrow C$

  then "as in propositional logic"

Predicate Logic - interpretations examples

D: natural numbers 0, 1, 2,...

+, ×: adition and multiplication

=: equality

$\mathbf{I} \models_v \exists y\, (y = x+1)$?

Predicate Logic - interpretations examples

D: natural numbers 0, 1, 2,...

+, ×: adition and multiplication

=: equality

$\mathbf{I} \models_{[0/x]} \exists y\, (y = x+1)$

$\mathbf{I} \not\models_{[0/x]} \exists y\, (x = y+1)$

Predicate Logic - interpretations examples

- D: integers ...-2, -1, 0, 1, 2,...

  +, ×: adition and multiplication

  =: equality

$\mathbf{I} \models_{[0/x]} \exists y\, (y = x+1)$

$\mathbf{I} \models_{[0/x]} \exists y\, (x = y+1)$

Predicate Logic -Truth and Validity

- A wwf A is *true* in an interpretation I iff every valuation in I satisfies A, *notation:* $I \models A$
- A wwf A is *false* in an interpretation I iff no valuation in I satisfies A

- A wwf A of a first order language L is (logically) *valid* iff it is true in every interpretation of L, *notation:* $\models A$
- A wwf A of a first order language L is (logically) *contradictory* iff it is false in every interpretation of L

Predicate Logic - interpretations examples

D:	natural numbers 0, 1, 2,...
+, ×:	adition and multiplication
=:	equality

$\mathbf{I} \models \forall x \exists y \, (y = x+1)$

$\mathbf{I} \not\models \forall x \exists y \, (x = y+1)$ since $\mathbf{I} \not\models_{[0/x]} \exists y \, (x = y+1)$

$\not\models \forall x \exists y \, (x = y+1)$ - follows from above!

$\not\models \forall x \exists y \, (y = x+1)$ - why?

Predicate Logic - quiz

Truth in N:	True	False	Valid	Contr.
1. $x+1 = y$				
2. $\forall x \, (x = x+1)$				
3. $\forall x \, \forall y \, (x+y = y+x)$				
4. $\exists x \, (P(x) \wedge \neg P(x))$				
5. $(\exists x \, \neg P(x)) \rightarrow (\neg \forall x \, P(x))$				

Predicate Logic - quiz

Truth in N:	True	False	Valid	Contr.
1. $x+1 = y$				
2. $\forall x \, (x = x+1)$		√		
3. $\forall x \, \forall y \, (x+y = y+x)$	√			
4. $\exists x \, (P(x) \wedge \neg P(x))$		√		√
5. $(\exists x \, \neg P(x)) \rightarrow (\neg \forall x \, P(x))$	√		√	

Predicate Logic -Truth and Validity

- Following Kelly we include the following predicate constants in our syntax for predicate logic:

- ⊥ standing for the always false predicate, i.e. the predicate which is false in every interpretation
- ⊤ standing for the always true predicate, i.e. the predicate which is true in every interpretation

Todays programme: Predicate Logic and Program Verification

- *Familiarity* with basic *concepts/results* of predicate logic
  - Syntax: variables, quantification, scope
  - Semantics: interpretations, valuations, satisfaction truth, validity
  - Axiomatic proof system FOPL
  - Gödels completeness theorem for predicate logic
- *Describe* the *use* of predicate logic in program verification
  - Syntax: program specifications, Hoare triples
  - Semantics: partial and total correctness
  - Proof system: Hoare proof rules

Predicate logic - axiomatic proof system

- Axioms:
  - Ax1 $A \rightarrow (B \rightarrow A)$
  - Ax2 $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
  - Ax3 $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$
- Deduction rules:
  - Modus ponens MP $\dfrac{A, \quad A \rightarrow B}{B}$

Predicate logic - axiomatic proof system

- Axioms:
  - Ax1 $A \rightarrow (B \rightarrow A)$
  - Ax2 $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
  - Ax3 $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$
  - Ax4 $(\forall x)\ A(x) \rightarrow A(t/x)$ where t is free for x in A!
  - Ax5 $(\forall x)\ (A \rightarrow B) \rightarrow (A \rightarrow (\forall x)\ B)$ no free occ's of x in A!
- Deduction rules:
  - Modus ponens MP $\dfrac{A, \quad A \rightarrow B}{B}$

Predicate logic - substitution

A[t/x] notation for
"A with all free occurrences of x substituted by t"

- Examples

$((\forall x\ (P(x) \wedge Q(x)) \rightarrow (P(x) \rightarrow Q(y)))\ [f(y)/x] =$

$(\forall x\ (P(x) \wedge Q(x)) \rightarrow (P(f(y)) \rightarrow Q(y))$

$((\forall y\ (P(y) \wedge Q(x)) \rightarrow (P(y) \rightarrow Q(x)))\ [f(y)/x] = ??$

Predicate logic - substitution

- A[t/x] is only defined if "t is free for x in A":
  no free occurrence of x in A occurs within the scope of $\forall y$ or $\exists y$ for any variable y occurring in t
- For all t,x,A, - t can always be made free for x in A by a suitable renaming of bindings $\forall y$, $\exists y$ in A
- Example

  $((\forall y\ (P(y) \wedge Q(x)) \rightarrow (P(y) \rightarrow Q(x)))\ [f(y)/x] =$

  $(\forall z\ (P(z) \wedge Q(f(y))) \rightarrow (P(y) \rightarrow Q(f(y)))$

Predicate logic - axiomatic proof system

- Axioms:
  - Ax1 $A \rightarrow (B \rightarrow A)$
  - Ax2 $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
  - Ax3 $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$
  - Ax4 $(\forall x)\ A(x) \rightarrow A(t)$ where t is free for x in A!
  - Ax5 $(\forall x)\ (A \rightarrow B) \rightarrow (A \rightarrow (\forall x)\ B)$ no free occ's of x in A!
- Inference rules:
  - Modus ponens MP $\dfrac{A, \quad A \rightarrow B}{B}$
  - Generalisation G $\dfrac{A}{(\forall x)\ A}$

Example of proof

- Assume that y does not occur in A(x)
  Prove $(\forall x)\ A(x) \rightarrow (\forall y)\ A(y)$

1.	$(\forall x)\ A(x)$	Hyp
2.	$(\forall x)\ A(x) \rightarrow A(y)$	Ax4 (y free for x in A)
3.	$A(y)$	MP 1,2
4.	$(\forall y)\ A(y)$	G

Pred. Logic - soundness and completeness

- **Gödel's Completeness Theorem**

  Our set of proof rules (the 3 axioms and MP from propositional logic plus the 2 extra axioms and G) is *sound* and *complete* for predicate logic!

- *Proof*

  Look for Gödel's proof!

Validity for predicate logic

- Validity problem for predicate logic:

  Given a first order predicate logic formula A,

  is A valid, i.e. |= A?

- **Theorem**

  The validity problem for predicate logic is unsolvable

  Proof: can be shown by a reduction from PCP

  - **Corollary**

    The set of valid formulas in predicate logic is recursively enumerable, but not recursive

  Proof: ??

Validity for predicate logic

- Validity problem for predicate logic:

  Given a first order predicate logic formula A,

  is A valid, i.e. |= A?

- **Theorem**

  The validity problem for predicate logic is unsolvable

  Proof: can be shown by a reduction from PCP

  - **Corollary**

    The set of valid formulas in predicate logic is recursively enumerable, but not recursive

  Proof: Gödel's completeness theorem

Todays programme: Predicate Logic and Program Verification

- *Familiarity* with basic *concepts/results* of predicate logic
  - Syntax: variables, quantification, scope
  - Semantics: interpretations, valuations, satisfaction truth, validity
  - Axiomatic proof system FOPL
  - Gödels completeness theorem for predicate logic
- *Describe* the *use* of predicate logic in program verification
  - Syntax: program specifications, Hoare triples
  - Semantics: partial and total correctness
  - Proof system: Hoare proof rules

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 example *C* = *Fac*

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


PLN semantics

- A PLN *state* associates natural numbers to program variables: *States: Var* → *N*
- The operational semantics of PLN defines the semantics of a program *C* as a PARTIAL function

  Sem[C]: States → *States*

  where *Sem[C](s)* =

s'	if *C* when started in state *s* terminates in state *s'*
undefined	otherwise

PLN semantics, example *C* = *Fac*

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

Sem[Fac](x = 4, y = 0, z = 0,...) =
(x= 4, y = 24, z = 4,...)

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
```

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

- $\models_{tot} \{ \top \} \; Fac \; \{ y = x! \}$

Pre/postcondition interpretation

- Let *N* be the predicate logic interpretation of natural numbers with a (yet unspecified) vocabulary of constants, functions and predicates - all interpreted ”in the standard way”.

- Note that PLN states are nothing but predicate logic valuations!

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 $N \models_{s'} \psi$

Hoare proof rules := and ;

$$\frac{}{\{\psi\,[E/x]\}\quad x := E \quad \{\psi\}}\ \text{Ass-axiom}$$

$$\frac{\{\phi\}\ C_1\ \{\eta\} \qquad \{\eta\}\ C_2\ \{\psi\}}{\{\phi\}\ C_1\,;\,C_2\ \{\psi\}}\ \text{Comp-rule}$$

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 ¬ (m = n) do

if m > n then m:=m-n

else n:= n-m;

r:= m

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

A proof of Euclid's gcd algorithm

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

while ¬ (m = n) do

if m > n then m:=m-n

else n:= n-m;

$\{\eta\}$

r:= m

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

A proof of Euclid's gcd algorithm

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

while $\neg$ (m = n) do

if m > n then m:=m-n

else n:= n-m;

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

r:= m

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

$$\frac{}{\{m = gcd(m_0, n_0)\}\ r := m\ \{r = gcd(m_0, n_0)\}}\ \text{Ass-axiom}$$

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

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

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

Hoare proof rules - implied

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

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 $\phi' \to \phi$ which are true in N - the interpretation of natural numbers!!!!

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:

$\vdash_N m = m_0 \geq 1 \wedge n = n_0 \geq 1$
-> $gcd(m,n) = gcd(m_0,n_0)$

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

Proofs using Hoare rules

- Notation:
  $\vdash_{par} \{ \phi \}\ C \{ \psi \}$ iff
  $\{ \phi \}\ C \{ \psi \}$ has a proof using the Hoare rules
  and rules for $\vdash_N$!!
- Are the Hoare rules sound and complete, i.e
  $\vdash_{par} \{ \phi \}\ C \{ \psi \}$ iff $\models_{par} \{ \phi \}\ C \{ \psi \}$???

Todays programme: Predicate Logic and Program Verification

- *Familiarity* with basic *concepts/results* of predicate logic
  - Syntax: variables, quantification, scope
  - Semantics: interpretations, valuations, satisfaction truth, validity
  - Axiomatic proof system FOPL
  - Gödels completeness theorem for predicate logic
- *Describe* the *use* of predicate logic in program verification
  - Syntax: program specifications, Hoare triples
  - Semantics: partial and total correctness
  - Proof system: Hoare proof rules

Exercises

- *Describe* the semantics of predicate logic
  - Kelly page 123 6.7 (scope rules)
  - Kelly page 130 6.9 (expressiveness
  - Kelly page 136 6.12 (satisfaction)
  - Kelly page 138 6.19 (satisfiability, truth, validity)
- *Describe* and *construct* deductions in FOPL
  - Kelly page 160 7.1 (i) (ii)
- *Describe* and *construct* deductions for Hoare triples
  - LimProVer page 10 Exercise 1