

- First Order Logic
 - Syntax
 - Binding and α Conversion
 - Omitting Parantheses - Binding Strengths
 - Semantics
 - Substitution
 - Universal and Existential Quantification
 - Equality
- Correctness
 - Termination
 - Well-founded Relations
 - Correctness - Behaviour
 - Correctness - equational reasoning
 - Correctness - reasoning by cases
 - Proof by Induction

First Order Logic

Syntax

There are two syntactic categories : **Terms** and **Formulae**.

Signature : consists of a set of function symbols \mathcal{F} and a set of Predicate symbols \mathcal{P} . We write f^k or (p^k) to indicate function symbol f or (predicate symbol p) has arity k . 0-arity function symbols are treated as constants.

Term, **the terms of first-order logic**, is the smallest set where

1. $x \in \text{Term}$ if $x \in \mathcal{V}$ and
2. $f^n(t_1, \dots, t_n) \in \text{Term}$ if $f^n \in F$ and $t_i \in \text{Term}$, for all $1 \leq i \leq n$

Form, **the formulae of first order logic**, is the smallest set where

1. $\perp \in \text{Form}$
2. $p^n(t_1, \dots, t_n) \in \text{Form}$ if $p^n \in \mathcal{P}$ and $t_j \in \text{Term}$, for all $1 \leq j \leq n$
3. $A \circ B \in \text{Form}$ if $A \in \text{Form}$, $B \in \text{Form}$, and $\circ \in \{\wedge, \vee, \rightarrow\}$
4. $Qx.A \in \text{Form}$ if $A \in \text{Form}$, $x \in \mathcal{V}$, and $Q \in \{\forall, \exists\}$

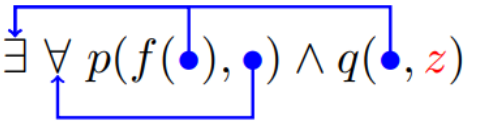
Each occurrence of each variable in a formula is **bound** or **free**

$$\forall x. \exists y. p(x, y) \equiv \forall y. \exists x. p(y, x)$$

A variable occurrence x in a formula A is **bound** if x occurs within a subformula B of A of the form $\exists x.B$ or $\forall x.B$ and is said to be **free** otherwise.

Binding and α Conversion

Note that names of bound variables are irrelevant, they just encode the binding structure.

$\exists x. \forall y. p(f(x), y) \wedge q(x, z)$ stands for 

We can **rename** bound variables at any time : **α conversion** but it must preserve binding structure.

Here's an example :

$$\forall x. \exists y. p(x, y) \rightarrow \forall y. \exists x. p(y, x)$$

Omitting Parantheses - Binding Strengths

For binary operators :

- \wedge binds stronger than \vee
- \vee binds stronger than \rightarrow
- \rightarrow associates to the right
- \wedge and \vee to the left
- \neg binds stronger than any other binary operator

Quantifiers i.e. \forall and \exists extend to the right as far as possible, that is either end of line or `)`.

Semantics

A **structure** is a pair $S = \langle \mathcal{U}_S, \mathcal{I}_S \rangle$ where \mathcal{U}_S is a nonempty set, the **universe**, and \mathcal{I}_S is a mapping where :

1. $\mathcal{I}_S(p^n)$ is an n -ary relation on \mathcal{U}_S for $p^n \in \mathcal{P}$, and
2. $\mathcal{I}_S(f^n)$ is an n -ary (total) function on \mathcal{U}_S , for $f^n \in \mathcal{F}$

As shorthand, we write p^S for $\mathcal{I}_S(p)$ and f^S for $\mathcal{I}_S(f)$.

An **Interpretation** is a pair $I = \langle \mathcal{S}, v \rangle$, where $\mathcal{S} = \langle \mathcal{U}_S, \mathcal{I}_S \rangle$ is a structure and $v : \mathcal{V} \rightarrow \mathcal{U}_S$ a valuation.

The **value** of a term t under the interpretation $\mathcal{I} = \langle \mathcal{S}, v \rangle$ is written as $\mathcal{I}(t)$ and defined by

1. $\mathcal{I}(x) = v(x)$ for $x \in \mathcal{V}$ and
2. $\mathcal{I}(f(t_1, \dots, t_n)) = f^S(\mathcal{I}(t_1), \dots, \mathcal{I}(t_n))$

Satisfiability : $\models \subseteq \text{Intrepretations} \times \text{Form}$ is the smallest relation satisfying

- $\langle \mathcal{S}, v \rangle \models \exists x. A$ if $\langle \mathcal{S}, v[x \rightarrow a] \rangle \models A$ for some $a \in \mathcal{U}_S$
- $\langle \mathcal{S}, v \rangle \models \forall x. A$ if $\langle \mathcal{S}, v[x \rightarrow a] \rangle \models A$ for all $a \in \mathcal{U}_S$

Here $v[x \rightarrow a]$ is the valuation v' identical to v , except that $v'(x) = a$

- When $\langle \mathcal{S}, v \rangle \models A$ we say **A is satisfied with respect to $\langle \mathcal{S}, v \rangle$** or $\langle \mathcal{S}, v \rangle$ is a **model** of A .
- When every suitable interpretation is a model, we write $\models A$ and say **A is valid**.
- A is **satisfiable** if there is at least one model for A (and contradictory otherwise)

Here's an example for a model:

$$\forall x. p(x, s(x))$$

- $\mathcal{U}_S = \mathcal{N}$

- $p^S = \{(m, n) | m, n \in U_S \text{ and } m < n\}$
- $s^S(x) = x + 1$

Substitution

As the name suggest we can replace in A all occurences of a free variable x with some term t . We write $A[x \rightarrow t]$ to indicate that we substitute x by t in A .

For $A = \exists y. y \cdot x = x \cdot z$ we can substitute :

$$A[x \rightarrow 2 - 1] = \exists y. y \cdot (2 - 1) = (2 - 1) \cdot z$$

Note that all free variables of t must still be free in $A[x \rightarrow t]$ i.e. *Avoid Capture*. If it deems necessary, α convert A before substitution.

Universal and Existential Quantification

Universal quantification :

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x. A} \forall - I^* \qquad \frac{\Gamma \vdash \forall x. A}{\Gamma \vdash A[x \rightarrow t]} \forall - E$$

with side condition $*$: x is not free in any assumption Γ

Existential Quantification :

$$\frac{\Gamma \vdash A[x \rightarrow t]}{\Gamma \vdash \exists x. A} \exists - I \qquad \frac{\Gamma \vdash \exists x. A \quad \Gamma, A \vdash B}{\Gamma \vdash B} \exists - E^*$$

with side condition $*$: x is neither free in B nor free in Γ

Equality

Note that we speak of *first-order logic with equality* rather than equality being "just another predicate". We add $t_1 = t_2 \in Form$ if $t_1, t_2 \in Term$

Equality is an equivalence relation shown in these rules

$$\frac{}{\Gamma \vdash t = t} ref \qquad \frac{\Gamma \vdash t = s}{\Gamma \vdash s = t} sym \qquad \frac{\Gamma \vdash t = s \quad \Gamma \vdash s = r}{\Gamma \vdash t = r} trans$$

Equality is also a congruence on terms and all definable relations

$$\frac{\Gamma \vdash t_1 = s_1 \quad \dots \quad \Gamma \vdash t_n = s_n}{\Gamma \vdash f(t_1, \dots, t_n) = f(s_1, \dots, s_n)} cong_1$$

$$\frac{\Gamma \vdash t_1 = s_1 \quad \dots \quad \Gamma \vdash t_n = s_n \quad \Gamma \vdash p(t_1, \dots, t_n)}{\Gamma \vdash p(s_1, \dots, s_n)} cong_2$$

Correctness

Two crucial properties we want our program to have are :

1. Termination i.e. important for some programs to terminate in a finite amount of time

2. Functional behavior i.e. the function should return correct value

Termination

If f is defined in terms of functions $g_1, \dots, g_k (g_i \neq f)$ and each g_i terminates, then so does f . Here is an example :

```
g x = x * x + 15
f x = (g x + x + 2) / 13
-- As all functions, i.e. g, f, +, *, / terminate and f is not recursive, f terminates
```

There is a sufficient condition for termination, that is when arguments are smaller along a well-founded order on the function's domain.

An order $>$ on a set S is **well founded** iff there is no infinite decreasing chain $x_1 > x_2 > x_3 \dots$ for $x_i \in S$. An example would be $>_{\mathbb{N}}$ since the natural numbers are bounded by 0 from the lower end. We write $>_S$ to indicate the domain S .

Well-founded Relations

We can construct new well-founded relations from existing ones using **composition**.

Let R_1 and R_2 be binary relations on a set S . The **composition** of R_1 and R_2 is defined as :

$$(a, c) \in R_2 \circ R_1 \iff \{(a, c) \in S \times S \mid \exists b \in S. aR_1b \wedge bR_2c\}$$

Transitive Closure : Let $R \subseteq S \times S$. We define the transitive closure as

$$R^+ = \bigcup_{n \geq 1} R^n \text{ where } R^{n+1} = R \circ R^n, \text{ for } n \geq 1$$

So aR^+b iff $aR^i b$ for some $i \geq 1$. You can also see it as "a path from a to b of length at least 1 in the graph of Relations".

Lemma : Let $R \subseteq S \times S$. Let $s_0, s_i \in S$ and $i \geq 1$. Then $s_0 R^i s_i$ iff there are $s_1, \dots, s_{i-1} \in S$ such that $s_0 R s_1, R \dots R s_{i-1} R s_i$

Theorem : If $>$ is a well-founded order on the Set S , then $>^+$ is also well-founded on S . $>^+$ has bigger steps than just one by one like $>$.

A few examples for Termination:

<pre>-- E1 f 0 = 0 f 1 = 1 f n = f (n-1) + f (n-2)</pre>	<pre>E2 g 0 = 1 g 1 = 1 g n = g (n+1) + g (n+2)</pre>	<pre>E3 h (0,y) = y h (x,y) = h (x-1, y+1)</pre>
--	---	--

E1 terminates as both recursive calls decrease and are well founded.

E2 does not terminate as only one recursive call decreases while the other increases, thus never reaching the basecase.

E3 terminates, unlike in E2, only the first argument needs to decrease to reach the basecase. It computes $x + y$.

Correctness - Behaviour

Lets look at the following two programs :

```
fac 0 = 1                fac2 (0,a) = a
fac n = n * fac (n-1)    fac2 (n,a) = fac2 (n-1, n*a)
```

We can test for a finite amount of numbers to get an idea whether they compute the saame or not. For that the *lambda* function is useful, although we will cover the lambda function later in the course.

```
-- Test for numbers 1 to 100
(\x -> fac x = fac2(x,y))[1..100]
```

Correctness - equational reasoning

One way to proof correctness is based on a simple idea : **functions are equations**. Let's look at the following example :

```
swap :: (Int, Int) -> (Int, Int)
swap (a,b) = (b,a)
```

This is formally equivalent to : $\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \text{swap}(a,b) = (b,a)$

In this case the proof follows the idea of $\text{swap}(\text{swap}(a,b)) = (a,b)$. More generally proofs in first-order logic with equality.

Correctness - reasoning by cases

As the title suggest another way to prove correctness is by covering each case that could occur. In the following example we want to prove `maxi n m >= n`

```
maxi :: Int -> Int -> Int
maxi n m
  | n >= m    = n
  | otherwise = m
```

We have

$n \geq m \vee \neg(n \geq m)$ this holds as $P \vee \neg P$ is always true for all Propositions (Excluded Middle)

Now we show `maxi n m >= n` for both cases.

Case 1 $n \geq m$

- then `maxi n m = n` and $n \geq n$

Case 2 $\neg(n \geq m)$

- then `maxi n m = m`. But $m > n$, so `maxi n m >= n`

This uses the \vee -E rule i.e. given $Q \vee R$ to prove any P we must prove : 1. P follows from Q and 2. P follows from R

Proof by Induction

The classic of the classic. We use the *domino principle* formulated by the *Induction proof rule*.

To prove $\forall n \in \mathbb{N}. P$

- **Base Case** : Prove $P[n \rightarrow 0]$
- **Step Case** : For an arbitrary m not free in P , prove $P[n \rightarrow m + 1]$ under the assumption $P[n \rightarrow m]$ for $m \geq 0$, equivalently, proving $P[n \rightarrow m]$ from $P[n \rightarrow m - 1]$ for $m \geq 0$

Here is an example that uses Induction to prove that two programs can output the same values under certain modifications.

```
-- computes 2^r as 2 * 2 * 2 .. * 1      -- computes 1 + 2 + 4 + ... + 2^r
power2 :: Int -> Int                    sumPowers :: Int -> Int
power2 0 = 1                            sumPowers 0 = 1
power2 r = 2 * power2 (r-1)             sumPowers r = sumPowers (r-1) + power2 r
```

Conjecture: $\forall n \in \mathbb{N}. (sumPowers n) + 1 = power2(n + 1)$

Proof: Let $P \equiv (sumPowers n) + 1 = power2(n + 1)$ We show $\forall n \in \mathbb{N}. P$ by induction on n .

Base Case : Show $P[n \rightarrow 0]$

$$\begin{aligned}(sumPowers 0) + 1 &= 1 + 1 = 2 \\ power2(0 + 1) &= 2 \cdot power2 0 = 2 \cdot 1 = 2\end{aligned}$$

Step Case : Assume $P[n \rightarrow m]$ for an arbitrary m (not in P) i.e.

$$(sumPowers m) + 1 = power2(m + 1)$$

and prove $P[n \rightarrow m + 1]$ i.e.

$$\begin{aligned}(sumPowers(m + 1)) + 1 &= power2((m + 1) + 1) \\ (sumPowers(m + 1)) + 1 &= sumPowers((m + 1) - 1) + power2(m + 1) + 1 && \text{(def.)} \\ &= sumPowers(m) + 1 + power2(m + 1) && \text{(arithmetic)} \\ &= power2(m + 1) + power2(m + 1) && \text{(ind. hypothesis)} \\ &= 2 \cdot power2(m + 1) && \text{(arithmetic)} \\ &= power2(m + 2) && \text{(def.)}\end{aligned}$$

Thus we have proven `(sumPowers n) + 1 = power2 (n+1)`

Note that Induction works on any **well-founded domain** $(S, <_S)$ not just $(\mathbb{N}, <)$ the natural numbers with the "smaller than" relation.

Structural Weak Induction form : sometimes it is of benefit to use the weak form of the induction

$\forall l \in \mathbb{N}. l < m \rightarrow P[n \rightarrow l]$ i.e. the ind. hypothesis holds for all l as long as they're $\leq m$