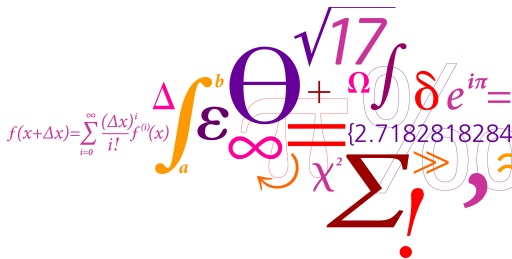


# 02157 Functional Programming

Lecture: Verification briefly

Michael R. Hansen



**DTU Compute**

Department of Applied Mathematics and Computer Science

You can prove properties of pure functional programs  
using “just”  
your prerequisites in discrete mathematics

- Unit testing
  - test examples – a few sanity checks
- ...
- Property-based testing
  - Validation of properties of programs
  - Program correctness supported by statistical information
- ...
- Verification
  - properties of program proved to be correct
  - Program correctness guaranteed by mathematical proofs

# Pure functional programs

A simple setting for verification of

- **terminating** functional programs  
excludes, for example, let rec  $f(x) = 1 + f(x)$
- **having no side-effects**

Reasoning is guided by

- equations based on program declarations
- mathematical properties like  $e + e = 2e$
- induction based on natural numbers and data types

The simple reasoning breaks down in the presence of side effects,  
where, for example,  $e + e = 2e$  does not necessary hold.

# A very, very simple example: factorial function

We prove  $\forall n \in \mathbb{N}. \text{fact } n = n!$ , where

```
let rec fact =
  function
  | 0 -> 1                                (* Case 1 *)
  | n -> n * fact (n-1)                  (* Case 2 *)
```

using the following **well-known** induction rule for natural numbers

1.  $P(0)$  base case
  2.  $\forall n. (P(n) \Rightarrow P(n+1))$  inductive step
- 
- $$\forall n. P(n)$$

What is  $P(n)$ ?

Base case. We must prove  $\text{fact } 0 = 0! = 1$ . Trivial.

Inductive step. Consider arbitrary  $n \in \mathbb{N}$ . We must establish

$$\underbrace{\underbrace{\text{fact } n = n!}_{\text{induction hypothesis}}}_{P(n)} \Rightarrow \underbrace{\text{fact}(n+1) = (n+1)!}_{P(n+1)}$$

## Very, very simple example cont'd

Assume the induction hypothesis:

$$\text{fact } n = n! \quad (\text{Ind.hyp.})$$

The inductive step is established by:

$$\begin{aligned} & \text{fact}(n+1) \\ = & (n+1) \cdot \text{fact } n && \text{Case 2, as } n+1 \neq 0 \\ = & (n+1) \cdot n! && \text{Ind.hyp.} \\ = & (n+1)! \end{aligned}$$

Hence  $\forall n \in \mathbb{N}. \text{fact } n = n!$  by the induction rule.

Simple induction and equational reasoning

The simple reasoning breaks down in the presence of side effects

## The declaration

```
type 'a list =  
  | Nil                                // Nil is written []  
  | Cons of 'a * 'a list              // Cons(x,xs) is written x::xs
```

denotes an inductive definition of lists (of type 'a)

- [] is a list
- if  $x$  is an element and  $xs$  is a list, then  $x :: xs$  is a list
- lists can be generated by above rules only

The following structural induction rule is therefore sound:

1.  $P([])$  base case
  2.  $\forall xs. \forall x. (P(xs) \Rightarrow P(x :: xs))$  inductive step
- 
- $$\forall xs. P(xs)$$

```
let rec (@) xs ys = match xs with
  | [] -> ys                                (* A1 *)
  | x::xs -> x::(xs @ ys);;                (* A2 *)

let rec len xs = match xs with
  | [] -> 0                                (* L1 *)
  | _::xs -> 1+len xs;;                    (* L2 *)
```

Property:

$$\forall xs. len(xs@ys) = len(xs) + len(ys)$$



# Proof of property by induction (I)

We prove:  $\forall xs. len(xs@ys) = len(xs) + len(ys)$  (1)

Let  $P(xs)$  be  $len(xs@ys) = len(xs) + len(ys)$

Base case: We must establish:  $P([])$ :

$$\begin{aligned} & len([]@ys) \\ = & len(ys) && A1 \\ = & 0 + len(ys) && Arith. \\ = & len([]) + len(ys) && L1 \end{aligned}$$

# Proof of property by induction (II)

Remember:  $P(xs)$  is  $len(xs@ys) = len(xs) + len(ys)$  ind. hyp.

Inductive step: Consider arbitrary  $xs$  and  $x$ .

Assume  $P(xs)$  the induction hypothesis.

We must establish  $P(x :: xs)$ :

$$len((x :: xs)@ys) = len(x :: xs) + len(ys)$$

Simple equational reasoning suffices:

$$\begin{aligned}
 & len((x :: xs)@ys) \\
 = & len(x :: (xs@ys)) && A2 \\
 = & 1 + len(xs@ys) && L2 \\
 = & 1 + (len(xs) + len(ys)) && ind.hyp. \\
 = & (1 + len(xs)) + len(ys) && Arith. \\
 = & len(x :: xs) + len(ys) && L2
 \end{aligned}$$

Using the structural induction rule we have established

$$\forall xs. len(xs@ys) = len(xs) + len(ys)$$

Reasoning about functional programs is "easy"

- no side effects
- inductively defined types (lists, trees, ...)

Topics from Program analysis, Model checking and Verification are studied in a variety of courses, e.g. 02141, 02143, 02156, 02242, 02244, 02245, 02246 introducing different theories and using highly advanced tools