02157 Functional Programming

Lecture 8: Verification

Michael R. Hansen

DTU Compute
Department of Applied Mathematics and Computer Science

Simple setting

A simple setting for verification of terminating functional programs having no side-effects

- induction on natural numbers
- inductively defined datatypes (such as lists)
- structural induction on lists

which covers a wide range of interesting programs.

A very, very simple example: factorial function

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

$$\frac{\begin{array}{lll} 1. & P(0) & \text{base case} \\ 2. & \forall n.(P(n) \Rightarrow P(n+1)) & \text{inductive step} \end{array}}{\forall n.P(n)}$$

What is $P(n)$?

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

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

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

Very, very simple example cont'd

Assume the induction hypothesis:

$$\mathit{fact}\ n = n! \qquad (\mathit{Ind.hyp.})$$

The inductive step is established by:

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

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

Simple induction and equational reasoning

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

A simple example concerning tail recursion

Iterative version of factorial function:

```
let rec factA(n,p) =
   match n with
   | 0 -> p                       (* Case 1 *)
   | _ -> factA(n-1,n*p)          (* Case 2 *)
```

Prove that for every natural number n and every p:

$$factA(n, p) = n! \cdot p$$

Advice: State the induction hypothesis explicitly.

Example: iterative factorial function

We prove $\forall n \in \mathbb{N}.\ \forall p \in \mathbb{N}. \mathit{factA}(n,p) = n! \cdot p$, where

```
let rec factA(n,p) =
   match n with
   | 0 -> p                         (* Case 1 *)
   | _ -> factA(n-1,n*p)            (* Case 2 *)
```

using the induction rule for natural numbers.

What is $P(n)$?

Let $P(n) : \forall p \in \mathbb{N}. \mathit{factA}(n,p) = n! \cdot p$.

Base case. We must prove $\forall p \in \mathbb{N}. \mathit{factA}(0,p) = 0! \cdot p$. Trivial.

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

$$\underbrace{\forall p \in \mathbb{N}. \mathit{factA}(n,p) = n! \cdot p}_{\text{induction hypothesis}: P(n)} \Rightarrow \underbrace{\forall p \in \mathbb{N}. \mathit{factA}(n+1,p) = (n+1)! \cdot p}_{P(n+1)}$$

Example cont'd

Assume the induction hypothesis:

$$\forall p' \in \mathbb{N}. \mathit{factA}(n, p') = n! \cdot p' \qquad (\mathit{Ind.hyp.})$$

We must establish: $\forall p \in \mathbb{N}. \mathit{factA}(n+1, p) = (n+1)! \cdot p$

Consider arbitrary $p \in \mathbb{N}$.

$$\begin{array}{lll} & \mathit{factA}(n+1, p) & \\ = & \mathit{factA}(n, (n+1) \cdot p) & \mathit{Case}\ 2, \mathit{as}\ n+1 \neq 0 \\ = & n! \cdot (n+1) \cdot p & \mathit{Ind.hyp.}, p' \mapsto (n+1) \cdot p \\ = & (n+1)! \cdot p & \end{array}$$

which establishes the inductive step.

Hence $\forall n \in \mathbb{N} \forall p \in \mathbb{N}. \mathit{factA}(n, p) = n! \cdot p$, by the induction rule.

Structural induction over lists

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:

$$\begin{array}{lll} 1. & P([]) & \text{base case} \\ 2. & \forall xs.\forall x.(P(xs) \Rightarrow P(x :: xs)) & \text{inductive step} \\ \hline & \forall xs.P(xs) & \end{array}$$

Example

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

let rec len = function | [] -> 0 | _::xs -> 1+len xs;;
```

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 $P([\,])$: $len([]@ys) = len(ys) = 0 + len(ys) = len([]) + len(ys)$

Inductive step: Consider arbitrary xs and x. Assume $P(xs)$.
We must establish $P(x :: xs)$:

$$\begin{array}{lll} & len((x :: xs)@ys) & \\ = & len(x :: (xs@ys)) & def.append \\ = & 1 + len(xs@ys) & def.len \\ = & 1 + (len(xs) + len(ys)) & ind.hyp. \\ = & (1 + len(xs)) + len(ys) & arith. \\ = & len(x :: xs) + len(ys) & def.len \end{array}$$

Using the structural induction rule we have established (1)

You can now solve problems like:

Prove

- `xs @ [] = xs`
- `[] @ ys = ys`
- `[x] @ ys = x::ys`
- `xs @ (ys @ zs) = (xs @ ys) @ zs`
- `naiveRev(xs @ ys) = naiveRev ys @ naiveRev xs`
- `revA(xs,ys) = naiveRev xs @ ys`

where `revA` and `naiveRev` are declared in the first part of the lecture.