Informatik and Mathematical Modelling 4/12 07
DTU Michael R. Hansen

Below are solutions to the functional part of the exam in 02153, 2006.

Problem 5

1.
```
Add(I "z",
    Let(D [("x", N 3), ("y",Add(I "x", N 1))],
        Add(I "x", I "y")));
```

2.
```
exception Env;

type env = (string * int) list;

fun lookup x [] = raise Env
  | lookup x ((y,v)::env) = if x=y then v else lookup x env;

fun update x v [] = [(x,v)]
  | update x v ((y,v')::env) = if x=y then (x,v)::env
                               else (y,v') :: update x v env;
```

3. including the solution to 4.

```
datatype expr =
  .....
  | ITE of bexpr * expr * expr (* question 4. *)
and bexpr =
    True
  | Le of expr * expr
  | And of bexpr * bexpr
and decl = D of (string * expr) list

(* E: expr -> env -> int *)
fun E e env =
   case e of
        N n          => n
      | I x          => lookup x env
      | Add(e1,e2)   => E e1 env + E e2 env
      | Sub(e1,e2)   => E e1 env - E e2 env
      | Let(d,e)     => E e (extend d env)
      | ITE(b,e1,e2) => if B b env then E e1 env else E e2 env
        (* question 4. *)
```

```
and B True _           = true                          (* question 4. *)
  | B(Le(e1,e2)) env  = E e1 env <= E e2 env
  | B(And(b1,b2)) env = B b1 env andalso B b2 env

(* extend decl -> env -> env *)
and extend(D []) env         = env
  | extend(D((x,e)::ds)) env = extend(D ds) (update x (E e env) env)
```

Problem 6

```
fun f x y = if x <= y then y :: f x (y-1) else [];

fun g h ([], _) = false
  | g h (_, []) = false
  | g h (x::xs, y::ys) = h(x,y)
                         orelse g h (x::xs, ys)
                         orelse g h (xs, y::ys);
```

1. f has the type `f : int -> int -> int list`.
   `f 1 0 = []` and `f 1 3 = [3,2,1]`.
   `f x y` computes the list `[y, y-1, y-2, ... , x]` (and the empty list `[]` if $y < x$).
2. g has the type `g : ('a * 'b -> bool) -> 'a list * 'b list -> bool`.
   `g h (xs, ys)` is true iff h(x,y) is true for some x in xs and some y in ys.
3. `g (fn (x,y) => x=y) : ''a list * ''a list -> bool`.
   `g (fn (x,y) => x=y) (xs,ys)` is true if xs and ys have a common element

Problem 7

We give three solutions. One using induction on natural numbers, one using structural induction on lists, and one using well-founded induction (which turns out to be the simplest).

Proof by induction on natural numbers.

We prove:

$$\forall k \forall xs.(\text{take}(k, xs) \ @ \ \text{drop}(k, xs) = xs)$$

Let $P(k)$ be $\forall xs.(\text{take}(k, xs) \ @ \ \text{drop}(k, xs) = xs)$.

Observe first that for any $k' \geq 0$ we have that

$$\text{take}(k', [\,])@\ \text{drop}(k', [\,]) = [\,] \qquad (*)$$

by the first clauses (t1) and (d1) of take and drop and the first clause of `@`.

The base case $P(0)$. Consider arbitrary xs. We must prove:

$$\text{take}(0, xs)@\ \text{drop}(0, xs) = xs$$

The case where $xs = [\,]$ is covered by $(*)$, so assume that $xs \neq [\,]$:

$$\begin{array}{lll} & \text{take}(0, xs) \ @ \ \text{drop}(0, xs) & \\ = & [\,]@\ xs & \text{by (t2, d2)} \\ = & xs & \text{by } \texttt{@1} \end{array}$$

For the inductive step we must prove:

$$\forall k.(P(k) \implies P(k+1))$$

Consider an arbitrary $k \geq 0$ and assume the induction hypothesis $P(k)$:

$$\forall xs'.(\text{take}(k, xs') \ @ \ \text{drop}(k, xs') = xs')$$

Consider an arbitrary xs. We must prove

$$\text{take}(k+1, xs) \ @ \ \text{drop}(k+1, xs) = xs$$

The case where $xs = [\,]$ is covered by $(*)$ so assume $xs \neq [\,]$. I.e. xs can be written in the form $xs = y :: ys$ and the inductive step is established by:

$$\begin{array}{lll} & \text{take}(k+1, y :: ys) \ @ \ \text{drop}(k+1, y :: ys) & \\ = & (y :: \text{take}(k, ys))@\ \text{drop}(k, ys) & \text{by (t3, d3)} \\ = & y :: (\text{take}(k, ys)@\ \text{drop}(k, ys)) & \text{by } (\texttt{@2}) \\ = & y :: ys & \text{using ind. hyp. with } xs' = ys \end{array}$$

and the proof is thereby completed.

Notice the use (and need for) the explicit quantifier $\forall xs'$ in the induction hypothesis when it is used above. This part is handled elegantly in a proof using well-founded induction.

Proof by induction on structural induction on lists.

We prove:

$$\forall xs \forall k.(\mathrm{take}(k, xs) \mathbin{@} \mathrm{drop}(k, xs) = xs)$$

In the following we assume that k and k' range over natural numbers.

Let $P(xs)$ be $\forall k.(\mathrm{take}(k, xs) \mathbin{@} \mathrm{drop}(k, xs) = xs)$.

The base case $P([\,])$. Consider arbitrary $k \geq 0$. We must prove:

$$\mathrm{take}(k, [\,]) @ \mathrm{drop}(k, [\,]) = [\,]$$

This follows easily by using the first clauses for take, drop and `@`.

For the inductive step we must prove:

$$\forall xs \forall x.(P(xs) \implies P(x :: xs))$$

Consider arbitrary list xs and element x of suitable types. Assume the induction hypothesis $P(xs)$:

$$\forall k'.(\mathrm{take}(k', xs) \mathbin{@} \mathrm{drop}(k', xs) = xs)$$

Consider an arbitrary $k \geq 0$. We must prove

$$\mathrm{take}(k, x :: xs) \mathbin{@} \mathrm{drop}(k, x :: xs) = x :: xs$$

Due to the form of the clauses for take and drop there are two cases to consider.

Case $k = 0$, which follows from:

$$\begin{array}{lll} & \mathrm{take}(0, x :: xs) \mathbin{@} \mathrm{drop}(0, x :: xs) & \\ = & [\,] @ \, (x :: xs) & \text{by (t2, d2)} \\ = & x :: xs & \text{by } \texttt{@} \end{array}$$

Case $k > 0$:

$$\begin{array}{lll} & \mathrm{take}(k, x :: xs) \mathbin{@} \mathrm{drop}(k, x :: xs) & \\ = & (x :: \mathrm{take}(k-1, xs)) @ \, \mathrm{drop}(k-1, xs) & \text{by (t3, d3)} \\ = & x :: (\mathrm{take}(k-1, xs) @ \, \mathrm{drop}(k-1, xs)) & \text{by } (\texttt{@}2) \\ = & x :: xs & \text{using ind. hyp. with } k' = k-1 \end{array}$$

and the proof is thereby completed.

Notice the use (and need for) the explicit quantifier $\forall k'$ in the induction hypothesis when it is used above. This part is handled elegantly in a proof using well-founded induction.

Proof by well-founded induction.

We prove:

$$\forall (k, xs) \in \mathbb{N} \times \texttt{'a list}.(\mathrm{take}(k, xs) \mathbin{@} \mathrm{drop}(k, xs) = xs)$$

by well-founded induction using the ordering $(k', xs') \prec (k, xs)$ iff $k' < k$.

Consider an arbitrary $(k, xs) \in \mathbb{N} \times \texttt{'a list}$ and assume that

$$\forall (k', xs') \prec (k, xs).(\mathrm{take}(k', xs') \mathbin{@} \mathrm{drop}(k', xs') = xs') \qquad (*)$$

We must establish

$$\mathrm{take}(k, xs) \mathbin{@} \mathrm{drop}(k, xs) = xs$$

There are three cases to consider:

Case 1: $xs = [\,]$. This case is established by using the first clauses for take, drop and `@`.

Case 2: $k = 0$ and $xs \neq [\,]$. This case is established by using the second clauses for take and drop, respectively, and the first clause for `@`.

Case 3: $k > 0$ and $xs = y :: ys$. This case follows from:

$$\begin{array}{lll}
 & \mathrm{take}(k, y :: ys) \mathbin{@} \mathrm{drop}(k, y :: ys) & \\
= & (y :: \mathrm{take}(k-1, ys)) \mathbin{@} \mathrm{drop}(k-1, ys) & \text{by (t3, d3)} \\
= & y :: (\mathrm{take}(k-1, ys) \mathbin{@} \mathrm{drop}(k-1, ys)) & \text{by (@2)} \\
= & y :: ys & \text{using } (*) \text{ and that } (k-1, ys) \prec (k, xs)
\end{array}$$

and the proof is thereby completed using the rule for well-founded induction.