Typed Printf via Delimited Continuations

Functional Pearl

Kenichi Asai

Ochanomizu University

April 16, 2010

Problem: writing a type-safe printf

```
sprintf ("Hello world!")
```
$\rightsquigarrow$ `"Hello world!"`

```
sprintf (% str ^ "world!")
```
`"Hello"`

$\rightsquigarrow$ `"Hello world!"`

```
sprintf (% str ^ " is " ^ % int)
```
`"x" 3`

$\rightsquigarrow$ `"x is 3"`

The type of the boxes depend on the first argument of `sprintf`.

$\Longrightarrow$ Do we need dependent types? – No! [Danvy'98]

Direct-style Solution

```
let str s = s
let int i = string_of_int i

let sprintf pattern = reset (fun () -> pattern ())
let % dir = shift (fun k -> fun x -> k (dir x))
```

```
    sprintf (fun () -> % str ^ "world!") "Hello"
 ⇝  reset (fun () -> % str ^ "world!") "Hello"
 ⇝  reset (fun () -> fun x -> str x ^ "world!") "Hello"
 ⇝  (fun x -> str x ^ "world!") "Hello"
 ⇝  str "Hello" ^ "world!"
 ⇝  "Hello" ^ "world!"
 ⇝  "Hello world!"
```

Direct-style Solution

```
let str s = s
let int i = string_of_int i

let sprintf pattern = reset (fun () -> pattern ())
let % dir = shift (fun k -> fun x -> k (dir x))
```

- `sprintf` delimits the context.
- `%` changes the type of the context to receive more arguments.
- The solution requires `shift` and `reset`, or CPS transformation.

```
sprintf (fun () -> % str ^ "world!") "Hello"
```

How can we use shift/reset?

- Emulation using `call/cc` + one mutable cell [Filinski'94]
- Scheme48 + shift/reset [Gasbichler&Sperber 2002]
- MinCaml + shift/reset [Masuko&A.2009]
- OCaml + delimited continuations [Kiselyov2010]

On-going work: caml light with shift and reset.

- Supports answer type polymorphism/modification.
- Will be my main programming language very soon.
- Not public yet.

PS. More on typed printf, see [HOSC'09].