

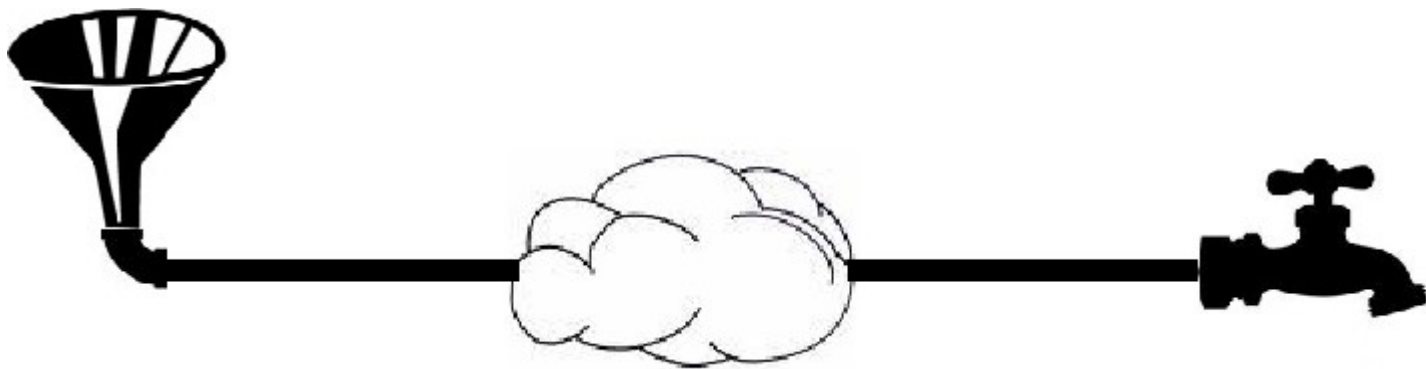
Qui sème la fonction récolte le tuyau typé

Didier Parigot, Bernard P. Serpette

Inria Sophia-Antipolis - Méditerranée

“World”

“Hello”

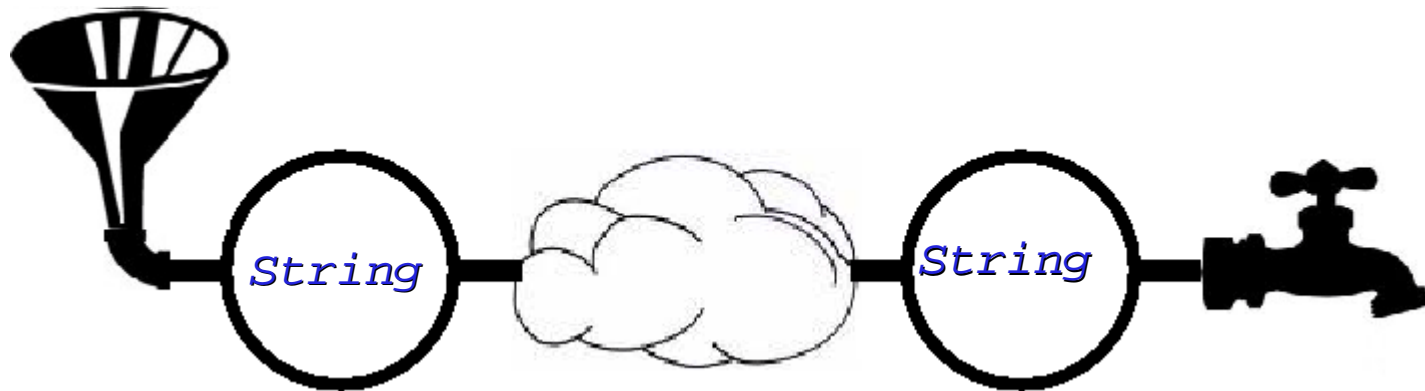


“World”

“Hello”

“World”

“Hello”



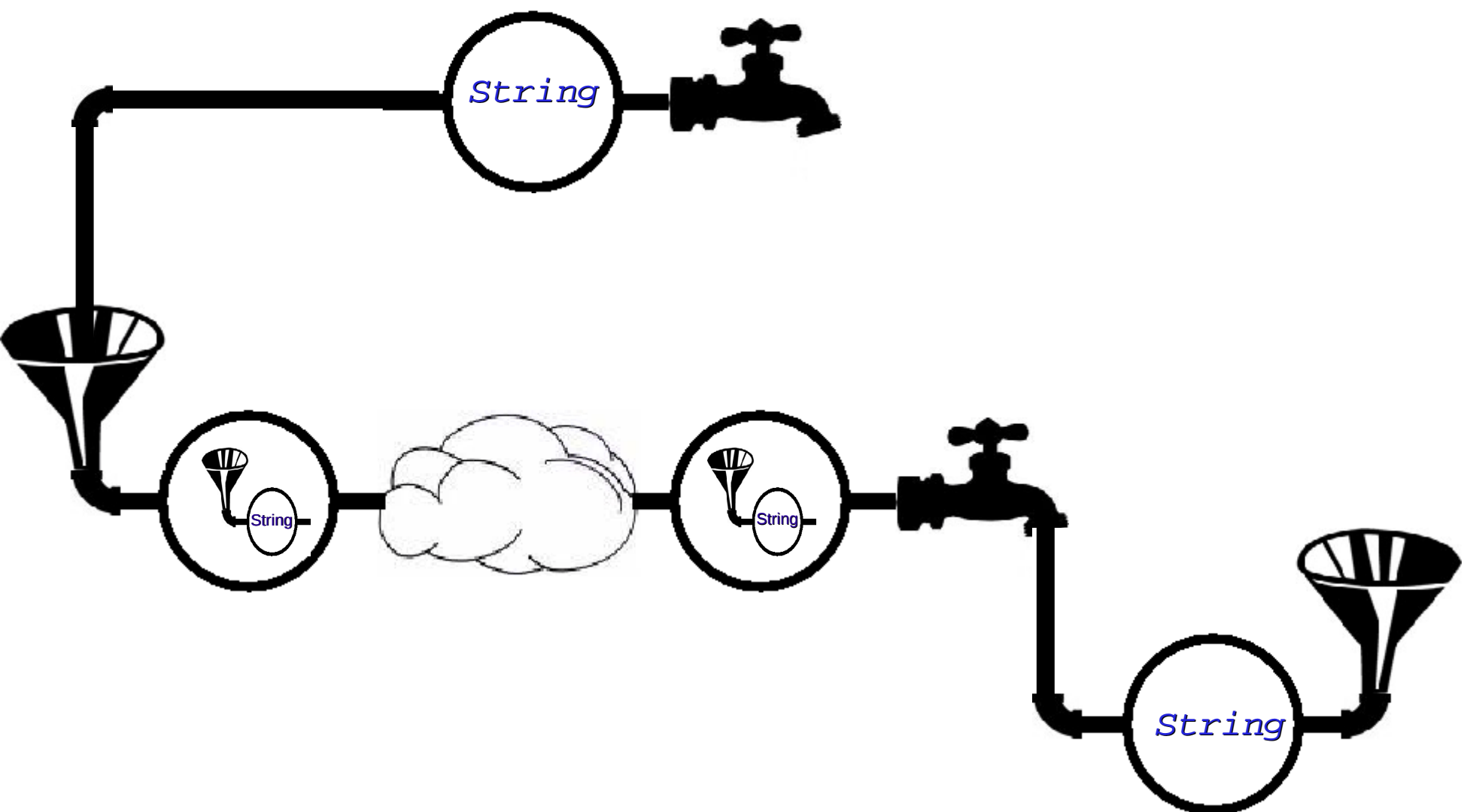
“! uiastropecrasecesarceportsalut”

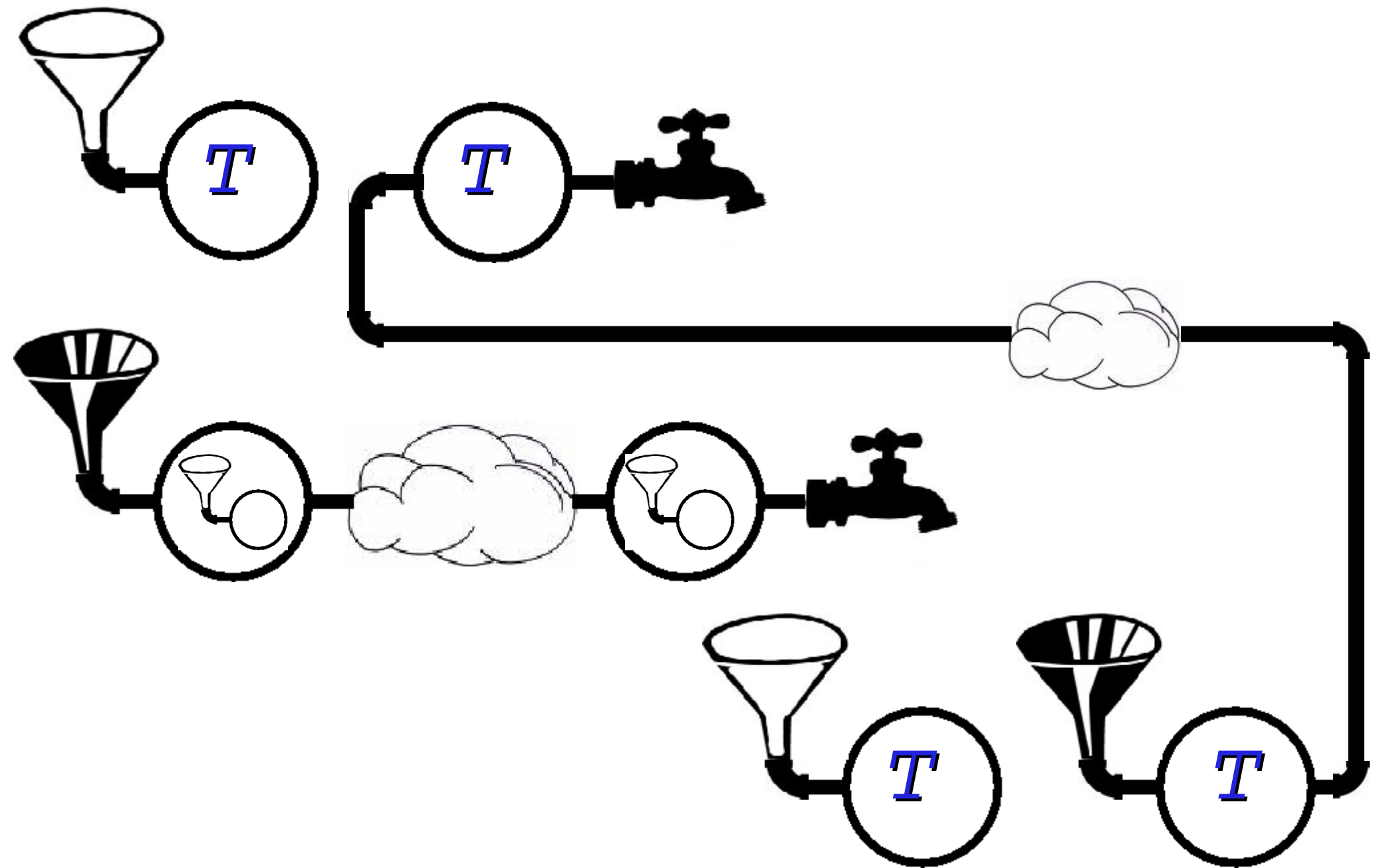
“Hello World”

“Hello”

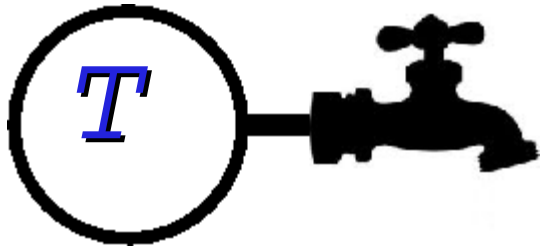
“Hello”

“World”

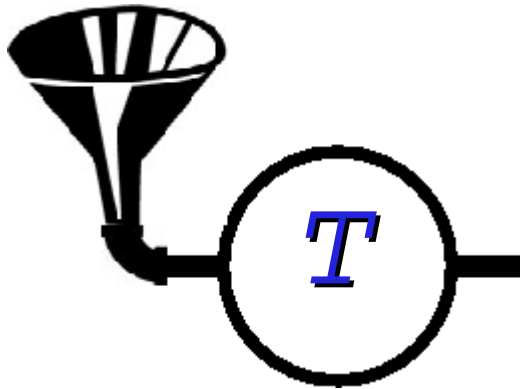




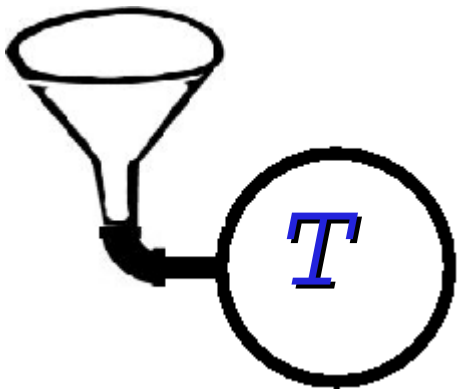
Type des protocoles



unit -> T



T -> unit



unit -> T -> unit

Un exemple de protocole

```
let qprot () =  
  let q = Queue.create () in  
  (fun () -> (Queue.pop q)),  
  (fun () -> (fun (v) -> (Queue.push v q))))
```

*unit -> (unit -> 'a) * (unit -> 'a -> unit)*

Réflexivité

```
let self prot v =  
  let (read,writer)=(prot ()) in  
    ((writer ()) v);  
  (read ())
```

*(unit -> (unit -> 'a) * (unit -> 'b -> 'c)) -> 'b -> 'a*

```
let id x = self qprot x  
'a -> 'a
```

Symétrie

```
let sym prot out connect =  
  let (read, writer) = (prot ()) in  
    (out (fun (s) -> (connect s (writer ()))));  
  read
```

```
(unit -> 'a * (unit -> 'b)) ->  
(('c -> 'd) -> 'e) ->  
(('c -> 'b -> 'd) ->  
'a
```

Application de la symétrie

let call out $f =$

((sym qprot out (fun s w -> (w (f s)))) ())

(('a -> unit) -> 'b) -> ('a -> 'c) -> 'c

Transitivité

let strans prot out *get connect* =

let (read,writer)=(prot ()) in

(out (fun (a) ->

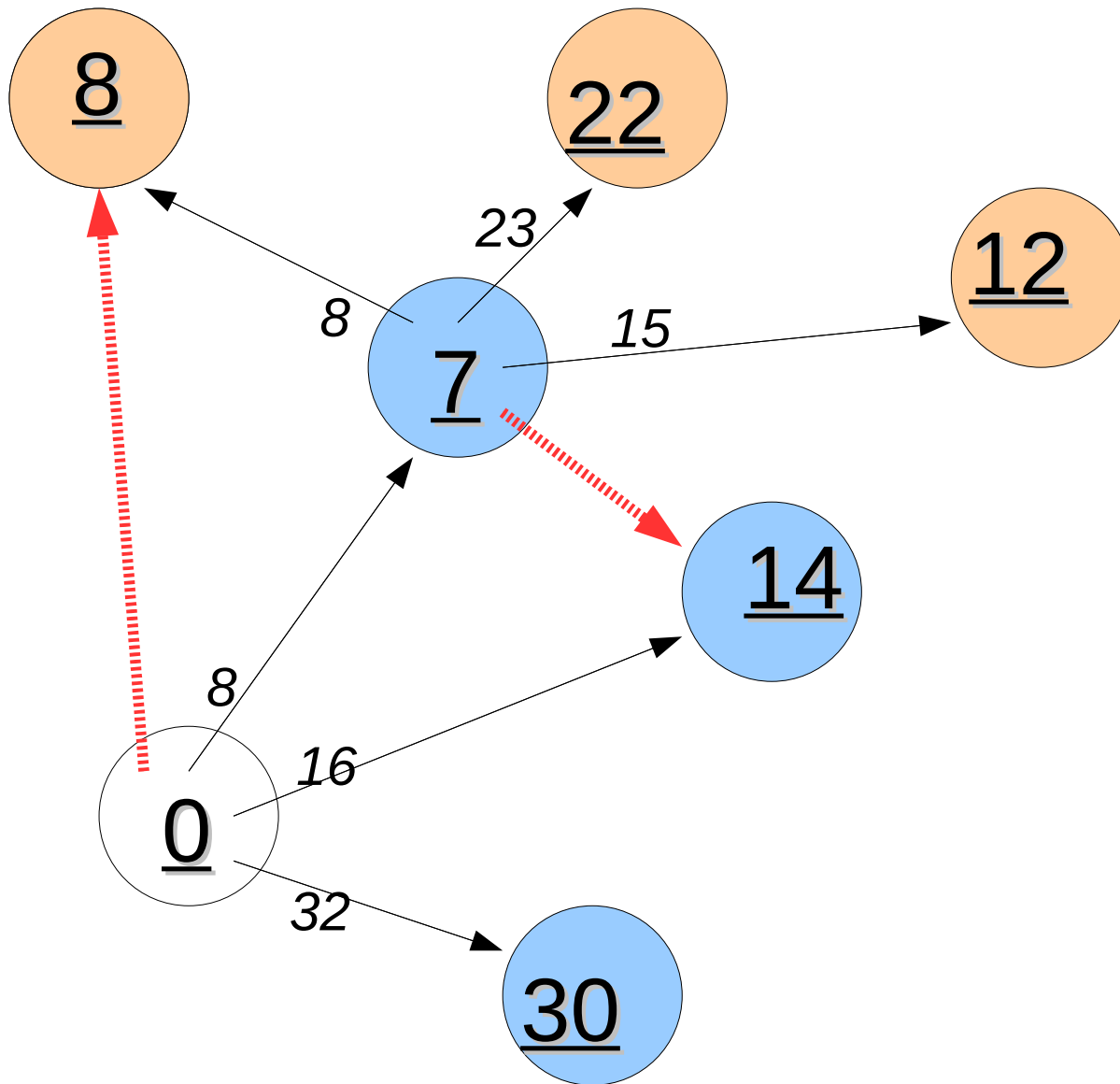
((*get* a)

(fun (b) -> (*connect* b (*writer* ())))));

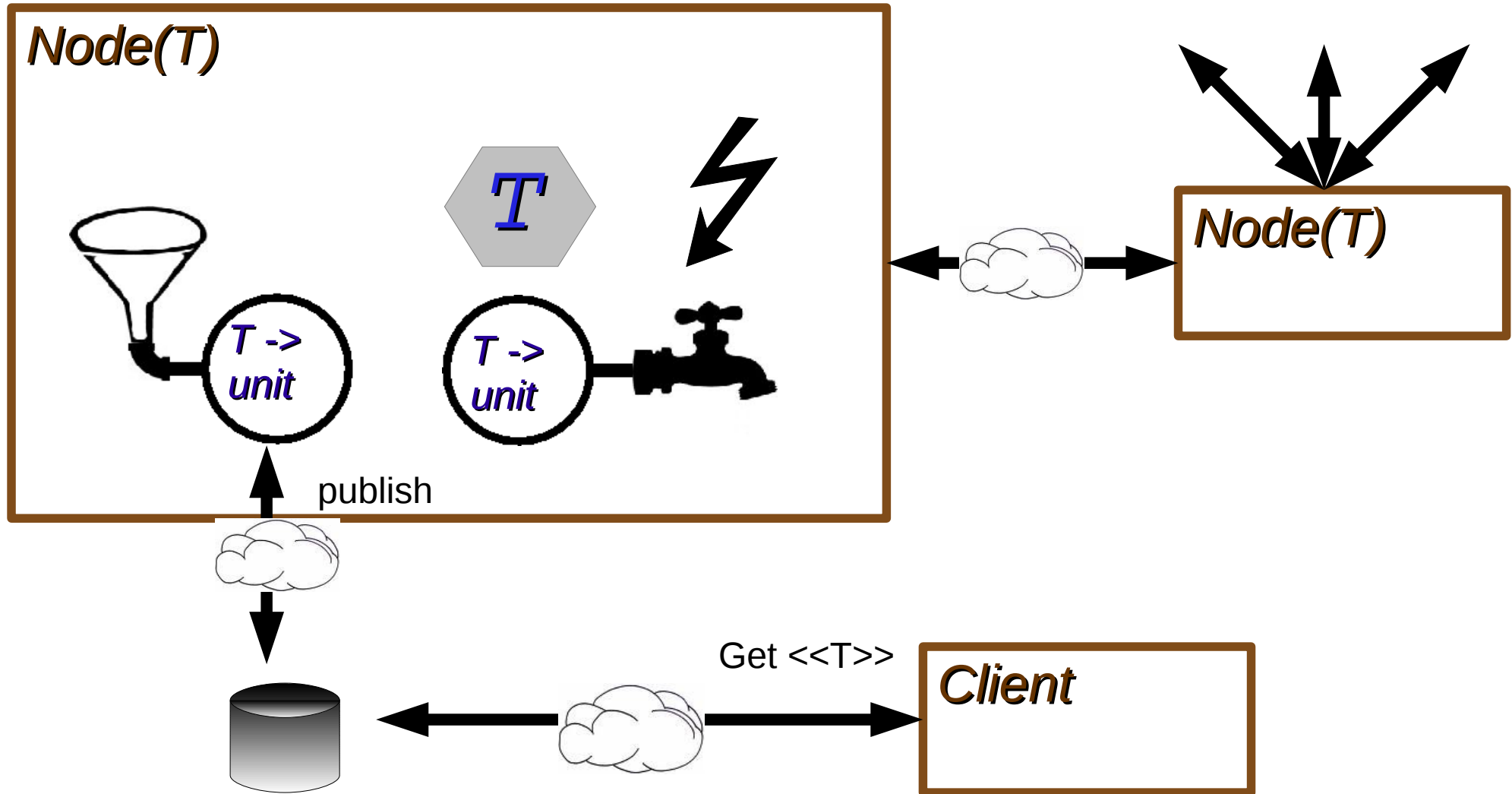
read

let sym p o c = strans p o (fun (a f) -> (f a)) c

Application de la transitivité



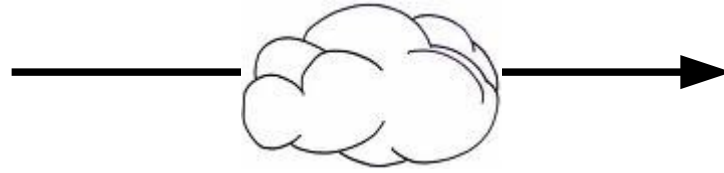
Bootstrap



$TRepr('a) \rightarrow (unit \rightarrow ('a \rightarrow unit) \rightarrow unit)$

Appel distant

A



fun x -> B

fun x -> B



Obj

let v = A in (fun f -> (f v))

fun x -> B

Qui sème la fonction ...

- ✓ ... s'abstrait du protocole de transport
- ✓ ... efface la génération de code (à la volée)
- ✓ ... récolte le tuyau typé
- x ... requiert une machine virtuelle
- x ... encourage le monolinguisme