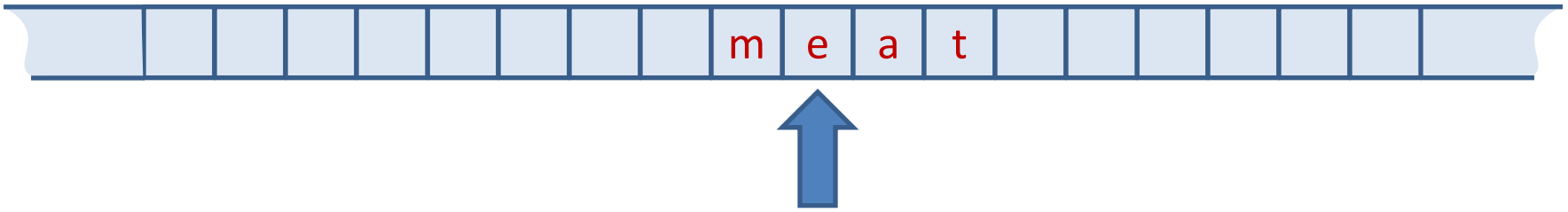


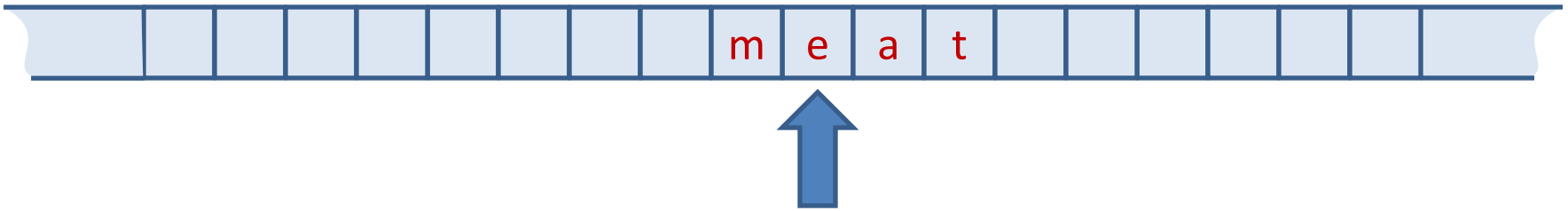
The Turing Machine Tape: module equivalence proof

COS 326

Speaker: Andrew Appel

Princeton University

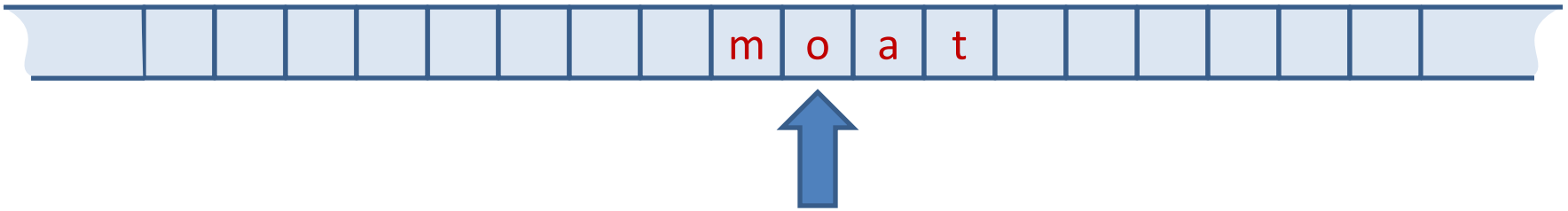
Module interface for Turing Machine Tape




```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

write ‘o’
left
read
write ‘b’

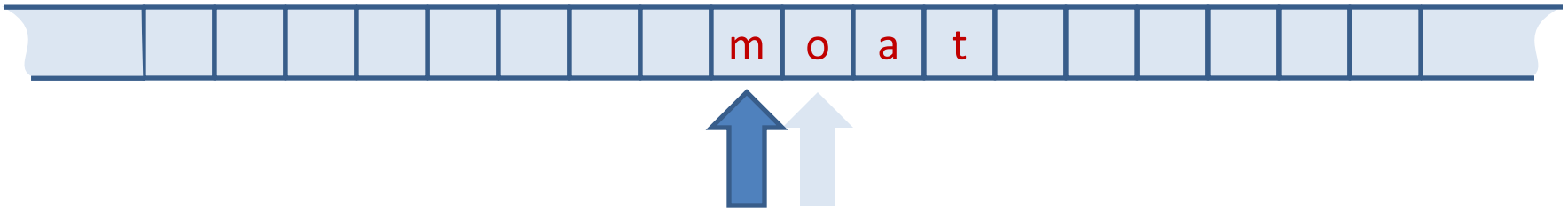
Module interface for Turing Machine Tape




```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

write ‘o’
left
read
write ‘b’

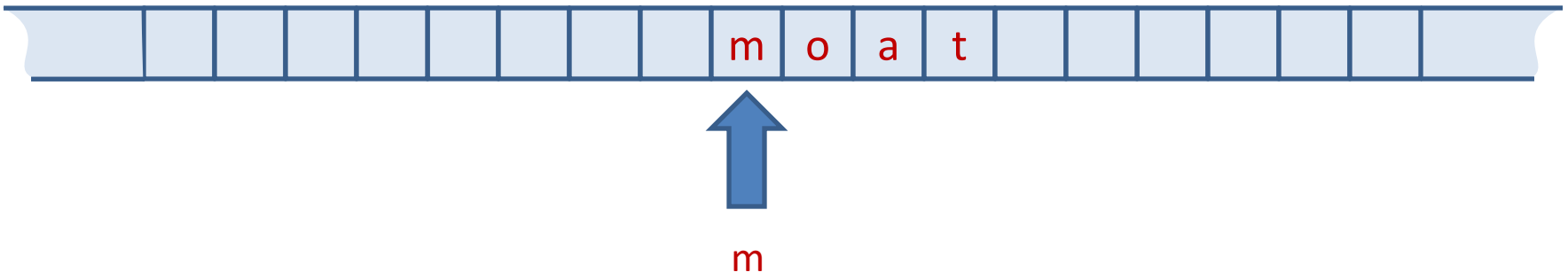
Module interface for Turing Machine Tape




```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

write ‘o’
left
read
write ‘b’

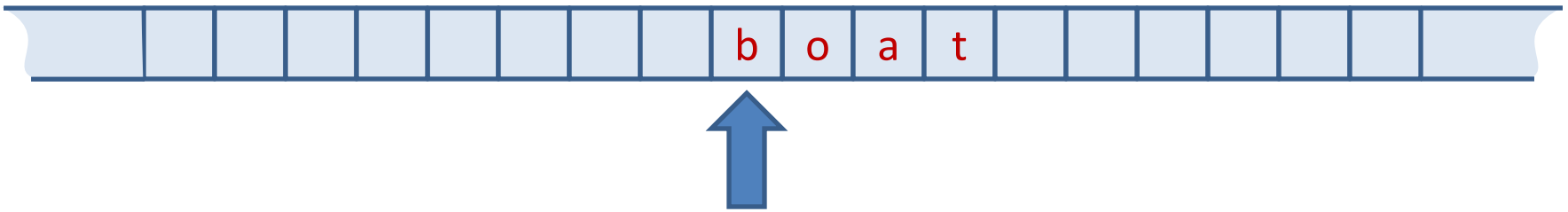
Module interface for Turing Machine Tape




```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

write ‘o’
left
read
write ‘b’

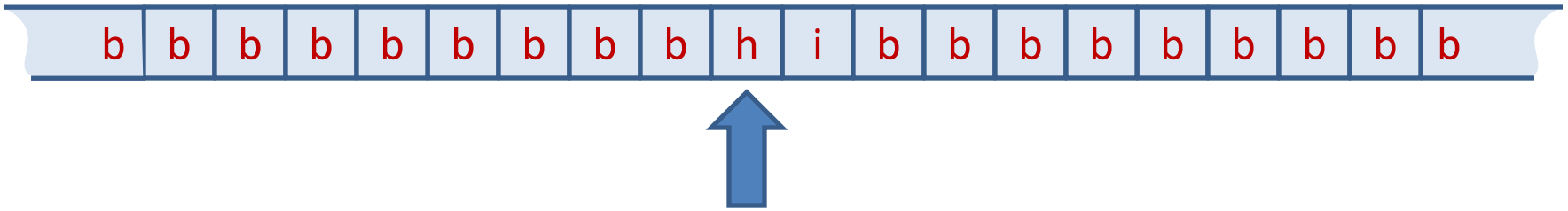
Module interface for Turing Machine Tape




```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

write ‘o’
left
read
write ‘b’

Module interface for Turing Machine Tape

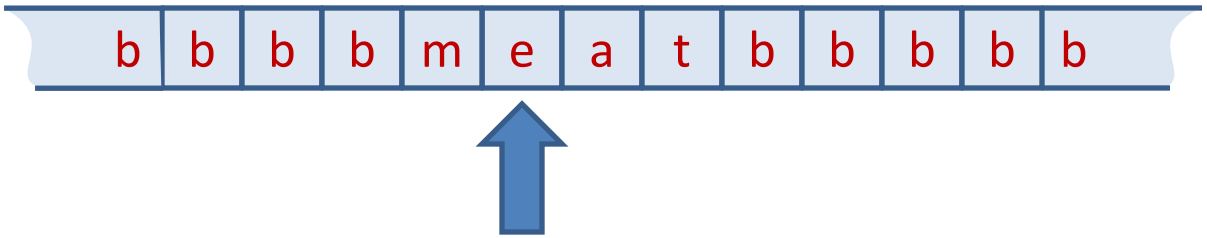



```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape : TAPE = struct ... end

let t = Tape.empty 'b'   in
let t = Tape.write t 'h'   in
let t = Tape.right t   in
let t = Tape.write t 'i'  in
let t = Tape.left t  in
t
```

A “reference implementation”




```
f(i) = match i with
       | -1 → m
       | 0 → e
       | 2 → a
       | 3 → t
       | _ → b
```

```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape0 : TAPE = struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x = fun i → if i=0 then x else t i
  let left t = . . .
  let right t = . . .
end
```

EXAM PROBLEM 1

1a. Write the definitions of **left** and **right** (in module Tape0).

1b. Prove: forall t, read (right (write (left t) x)) = read t.

1c. Prove: There exists a tape t and symbol x such that
left (write t x) ≠ write (left t) x

```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape0 : TAPE = struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x = fun i → if i=0 then x else t i
  let left t =    write me!
  let right t =   write me!
end
```

Stop the video now and

SOLVE PROBLEM 1 BEFORE CONTINUING

SOLUTION TO PROBLEM 1a

1a. Write the definitions of **left** and **right** (in module Tape0).

1b. Prove: forall t, read (right (write (left t) x)) = read t.

1c. Prove: There exists a tape t and symbol x such that
left (write t x) ≠ write (left t) x

```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape0 : TAPE = struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x = fun i → if i=0 then x else t i
  let left t =     fun i → t(i−1)
  let right t =    fun i → t(i+1)
end
```

SOLUTION TO PROBLEM 1b

1a. Write the definitions of **left** and **right** (in module Tape0).

1b. Prove: forall t, read (right (write (left t) x)) = read t.

1c. Prove: There exists a tape t and symbol x such that
left (write t x) ≠ write (left t) x

```
  read (right (write (left t) x))
= (right (write (left t) x)) 0                    (eval)
= (fun i →(write (left t) x)(i+1))0               (eval)
= (write (left t) x) (0+1)                        (eval)
= write (left t)  x  1                            (math)
= (fun i → if i=0 then x else left t i) 1         (eval)
= (if 1=0 then x else left t 1)                   (eval)
= left t 1                                        (eval)
= (fun i → t(i−1)) 1                              (eval)
= t (1-1)                                         (eval)
= t 0                                             (eval)
= read t                                          (reverse eval)
```

```
module Tape0 : TAPE = struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x =
     fun i → if i=0 then x else t i
  let left t =    fun i → t(i−1)
  let right t =  fun i → t(i+1)
end
```

SOLUTION TO PROBLEM 1c

1c. Prove: There exists a tape t and symbol x such that
left (write t x) ≠ write (left t) x

Proof: if f=g then ∀i. f(i)=g(i).
Conversely, if there exists any i such that f(i) ≠g(i), then f ≠g.

t=(fun _ → b) x=a i=6

left (write t x) i
= left (write t x) 6
= (fun i → (write t x) (i−1)) 6
= write t x (6-1)
= write t x 5
= (fun i → if i=0 then x else t i) 5
= if 5=0 then x else t 0
= t 0
= (fun _ → b) 0
= b

write (left t) x i
= (fun i → if i=0 then x else left t i) 6
= if 6=0 then x else left t 6
= left t 6
= t (6-1)
= t 5
= (fun _ → b) 5
= b

t

let left t = fun i → t(i−1)
let right t = fun i → t(i+1)
end

OOPS, we're trying to prove they're ***not*** equal!

SOLUTION TO PROBLEM 1c

1c. Prove: There exists a tape t and symbol x such that
left (write t x) ≠ write (left t) x

Proof: if f=g, then for all i, f(i)=g(i).
Conversely, if there exists any i such that f(i) ≠g(i), then f ≠g.

t=(fun _ → b) x=a i=0

left (write t x) i
= left (write t x) 0
= (fun i → (write t x) (i—1)) 0
= write t x (0-1)
= write t x (-1)
= (fun i → if i=0 then x else t i) (-1)
= if -1=0 then x else t 0
= t 0
= (fun _ → b) 0
= b

write (left t) x i
= (fun i → if i=0 then x else left t i) 0
= if 0=0 then x else left t 0
= x
= a

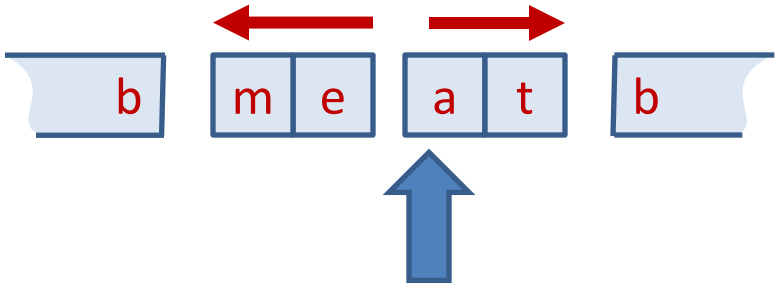
t

let write t x =
fun i → if i=0 then x else t i
let left t = fun i → t(i—1)
let right t = fun i → t(i+1)
end

b ≠ a, Q.E.D.

PROBLEM 2

An efficient implementation

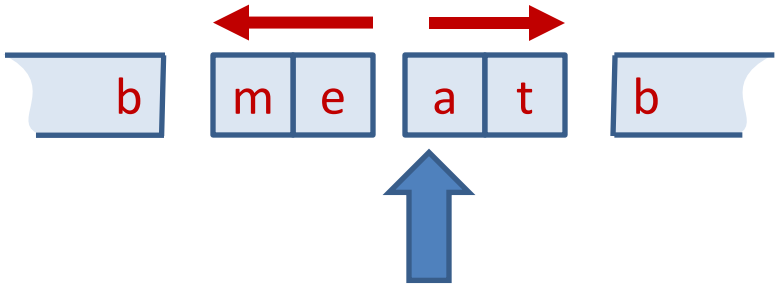


(b, [e;m], [a,t])

```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape1 : TAPE = struct
  type 'a t = 'a * 'a list * 'a list
  let empty b : 'a t = (b,[ ],[ ])
  let read (b,l,r) =
    (match r with [ ] → b | x::xs → x)
  let write (b,l,r) x =
     match r with [ ] → (b,l,[x]) _::xs → (b,l,x::xs)
  let left (b,l,r) =
    match l with x::xs → (b,xs,x::r) | [] → b,[],b::r)
  let right (b,l,r) =
    match r with x::xs → (b,x::l,xs) | [ ] → (b,b::l,r)
end
```

An efficient implementation



~~(b, [e;m], [a,t])~~

(b, [a;e;m], [t])

```
module type TAPE = sig
  type 'a t
  val empty: 'a → 'a t
  val read: 'a t → 'a
  val write: 'a t → 'a → 'a t
  val left: 'a t → 'a t
  val right: 'a t → 'a t
end
```

```
module Tape1 : TAPE = struct
  type 'a t = 'a * 'a list * 'a list
  let empty b : 'a t = (b,[ ],[ ])
  let read (b,l,r) =
    (match r with [ ] → b | x::xs → x)
  let write (b,l,r) x =
     match r with [ ] → (b,l,[x]) _::xs → (b,l,x::xs)
  let left (b,l,r) =
    match l with x::xs → (b,xs,x::r) | [] → b,[],b::r)
  let right (b,l,r) =
    match r with x::xs → (b,x::l,xs) | [ ] → (b,b::l,r)
end
```

An efficient implementation

Problem 2: Prove that Tape1 correctly implements a Turing Machine Tape, that is, Tape1 is *equivalent* to Tape0.

2a: Show an *abstraction function* R: 'a Tape1.t → 'a Tape0.t that relates values of type 'a Tape1.t to values of type 'a Tape0.t.

```
module Tape0 : TAPE =
struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x =
    fun i → if i=0 then x else t i
  let left t = fun i → t(i−1)
  let right t = fun i → t(i+1)
end
```

```
module Tape1 : TAPE = struct
  type 'a t = 'a * 'a list * 'a list
  let empty b : 'a t = (b,[ ],[ ])
  let read (b,l,r) =
    (match r with [ ] → b | x::xs → x)
  let write (b,l,r) x =
     match r with [ ] → (b,l,[x]) _::xs → (b,l,x::xs)
  let left (b,l,r) =
    match l with x::xs → (b,xs,x::r) | [] → (b,[],b::r)
  let right (b,l,r) =
    match r with x::xs → (b,x::l,xs) | [ ] → (b,b::l,r))
end
```

An efficient implementation

Problem 2: Prove that Tape1 correctly implements a Turing Machine Tape, that is, Tape1 is *equivalent* to Tape2.

2a: Show an *abstraction function* R : 'a Tape1.t → 'a Tape0.t that relates values of type 'a Tape1.t to values of type 'a Tape0.t.

2b: Using R, for each operator, state what theorem must be proved:
2b(empty) 2b(read) 2b(write) 2b(left) 2b(right).

2c: Prove each of those four theorems:
2c(empty) 2c(read) 2c(write) 2c(left) ~~2c(right)~~.
You can leave out 2c(right) because it's so similar to 2c(left).

Stop the video now and

SOLVE PROBLEM 2 BEFORE CONTINUING

(This might take a couple of hours)

Solution to Problem 2a

```
let rec nth (default: 'a) (al: 'a list) (i: int) =
  match al with [ ] → default  | a::r → if i=0 then a else nth default (i-1) r

let R ((b,l,r): 'a Tape1.t) : 'a Tape0.t =
 fun i → if i<0 then nth b l (-(i+1)) else nth b r i
```

```
module Tape0 : TAPE =
struct
  type 'a t = int → 'a
  let empty b : 'a t = fun i → b
  let read t = t 0
  let write t x =
    fun i → if i=0 then x else t i
  let left t = fun i → t(i—1)
  let right t = fun i → t(i+1)
end
```

```
module Tape1 : TAPE = struct
  type 'a t = 'a * 'a list * 'a list
  let empty b : 'a t = (b,[ ],[ ])
  let read (b,l,r) =
    (match r with [ ] → b | x::xs → x)
  let write (b,l,r) x =
     match r with [ ] → (b,l,[x]) _::xs → (b,l,x::xs)
  let left (b,l,r) =
    match l with x::xs → (b,xs,x::r) | [] → (b,[],b::r)
  let right (b,l,r) =
    match r with x::xs → (b,x::l,xs) | [ ] → (b,b::l,r))
end
```

Solution to Problem 2b

2b(empty): $\forall x.\ \ R\ (\text{Tape1.empty}\ x) = \text{Tape0.empty}\ x$

2b(read): $\forall t.\ \ \text{Tape1.read}\ t = \text{Tape0.read}\ (R\ t)$

2b(write): $\forall t.\ \forall x.\ \ R(\text{Tape1.write}\ t\ x) = \text{Tape0.write}\ (R\ t)\ x$

2b(left): $\forall t.\ \ R(\text{Tape1.left}\ t) = \text{Tape0.left}\ (R\ t)$

2b(right): $\forall t.\ \ R(\text{Tape1.right}\ t) = \text{Tape0.right}\ (R\ t)$

If you haven't done these proofs yet,

SOLVE PROBLEM 2C BEFORE CONTINUING

(This might take a couple of hours)

Solution to Problem 2c(empty)

2c(empty): $\forall x.\ R\ (Tape1.empty\ x) = Tape0.empty\ x$

To prove f=g, prove $\forall i.\ f(i)=g(i)$

R(Tape1.empty x) i

= R (x,[ ], [ ]) i

= (fun i → if i<0 then nth x [] (-(i+1)) else nth x [] i) I

= if i<0 then nth x [] (-(i+1)) else nth x [] i)

= if i<0 then x else x

= x

= (fun _ → x) i

= (Tape0.empty x) i

Solution to Problem 2c(read)

2c(read): $\forall t.\ \text{Tape1.read}\ t = \text{Tape0.read}\ (R\ t)$

Suppose $t=(b,l,r)$

$\text{Tape1.read}\ t$

$= \text{Tape1.read}\ (b,l,r)$

$= \text{match}\ r\ \text{with}\ [\,] \rightarrow b \mid x::xs \rightarrow x$

$= \text{match}\ r\ \text{with}\ [\,] \rightarrow b \mid a::r' \rightarrow a$

$= \text{match}\ r\ \text{with}\ [\,] \rightarrow b \mid a::r' \rightarrow \text{if}\ 0=0\ \text{then}\ a\ \text{else nth}\ b\ (i-1)\ r'$

$= \text{nth}\ b\ r\ 0$

$= \text{if}\ 0<0\ \text{then nth}\ b\ l\ (-(i+1))\ \text{else nth}\ b\ r\ 0$

$= (\text{fun}\ i \rightarrow \text{if}\ i<0\ \text{then nth}\ b\ l\ (-(i+1))\ \text{else nth}\ b\ r\ i)\ 0$

$= R\ (b,l,r)\ 0$

$= \text{Tape0.read}\ (R\ (b,l,r))$

$= \text{Tape0.read}\ (R\ t)$

Solution to Problem 2c(write)

2c(write): $\forall t.\ \forall x.\ R(\text{Tape1.write } t\ x) = \text{Tape0.write } (R\ t)\ x$

that is, $\forall t.\ \forall x.\ \forall i.\ R(\text{Tape1.write } t\ x)\ i = \text{Tape0.write } (R\ t)\ x\ i$

Suppose t=(b,l,r)

R (Tape1.write t x) i

= R (Tape1.write (b,l,r) x) i

= R ((match r with [ ] → (b,l,[x]) | _::ys → (b,l,x::ys)) i

=

Case Analysis! Case 1: r=[ ]

Solution to Problem 2c(write)

2c(write): $\forall t.\ \forall x.\ R(\text{Tape1.write } t\ x) = \text{Tape0.write } (R\ t)\ x$

that is, $\forall t.\ \forall x.\ \forall i.\ R(\text{Tape1.write } t\ x)\ i = \text{Tape0.write } (R\ t)\ x\ i$

Suppose t=(b,l,r)

Case 1: r=[ ]

R (Tape1.write t x) i

= R (Tape1.write (b,l,r) x) i

= R ((match r with [ ] → (b,l,[x]) | _::ys → (b,l,x::ys)) i

= R (b,l,[x]) i

= if i<0 then nth b l (-(1+i)) else nth b [x] i

= if i=0 then x else if i<0 then nth b l (-(i+1)) else nth b r i

= if i=0 then x else R t i

= Tape0.write (R t) x i

Solution to Problem 2c(write)

2c(write): $\forall t.\ \forall x.\ R(\text{Tape1.write } t\ x) = \text{Tape0.write } (R\ t)\ x$

that is, $\forall t.\ \forall x.\ \forall i.\ R(\text{Tape1.write } t\ x)\ i = \text{Tape0.write } (R\ t)\ x\ i$

Suppose t=(b,l,r)

Case 1: r=[ ]

R (Tape1.write t x) i

= R (Tape1.write (b,l,r) x) i

= R ((match r with [ ] → (b,l,[x]) | _::ys → (b,l,x::ys)) i

= R (b,l,[x]) i

= if i<0 then nth b l (-(1+i)) else nth b [x] i

= if i<0 then nth b l (-(1+i)) else

match [x] with [ ] → default | a::r → if i=0 then a else nth b (i-1) r

= if i<0 then nth b l (-(1+i)) else if i=0 then x else nth b [ ] (i-1)

= if i=0 then x else if i<0 then nth b l (-(i+1)) else nth b [ ] i

= if i=0 then x else R t i

= Tape0.write (R t) x i

Solution to Problem 2c(write)

2c(write): $\forall t.\ \forall x.\ R(\text{Tape1.write } t\ x) = \text{Tape0.write } (R\ t)\ x$

that is, $\forall t.\ \forall x.\ \forall i.\ R(\text{Tape1.write } t\ x)\ i = \text{Tape0.write } (R\ t)\ x\ i$

Suppose t=(b,l,r)

Case 2: r=(a::s)

R (Tape1.write t x) i

= R (Tape1.write (b,l,r) x) i

= R ((match a::s with [ ] → (b,l,[x]) | _::ys → (b,l,x::ys)) i

= R (b,l,x::s) i

= if i<0 then nth b l (-(i+1)) else nth b (x::s) i

= if i<0 then nth b l (-(i+1)) else if i=0 then x else nth b s (i-1)

= if i=0 then x else if i<0 then nth b l (-(i+1)) else nth b (a::s) i

= if i=0 then x else R t i

= Tape0.write (R t) x i

Solution to Problem 2c(write)

2c(write): $\forall t.\ \forall x.\ R(\text{Tape1.write } t\ x) = \text{Tape0.write } (R\ t)\ x$

that is, $\forall t.\ \forall x.\ \forall i.\ R(\text{Tape1.write } t\ x)\ i = \text{Tape0.write } (R\ t)\ x\ i$

Suppose t=(b,l,r)

Case 2: r=(a::s)

R (Tape1.write t x) i

= R (Tape1.write (b,l,r) x) i

= R ((match a::s with [ ] → (b,l,[x]) | _::ys → (b,l,x::ys)) i

= R (b,l,x::s) i

= if i<0 then nth b l (-(i+1)) else nth b (x::s) i

= if i<0 then nth b l (-(i+1)) else if i=0 then x else nth b s (i-1)

= if i=0 then x else if i<0 then nth b l (-(i+1)) else nth b s (i-1)

= if i=0 then x else if i<0 then nth b l (-(i+1)) else nth b (a::s) i

= if i=0 then x else R t i

= Tape0.write (R t) x i

Solution to Problem 2c(left)

You still have a chance to do this proof now, if you didn't do it yet . . .

Solution to Problem 2c(left)

2c(left): $\forall t.\ R\ (Tape1.left\ t) = Tape0.left\ (R\ t)$

so we will prove: $\forall t.\ \forall i.\ R\ (Tape1.left\ t)\ i = Tape0.left\ (R\ t)\ i$

Suppose t=(b,l,r)

R (Tape1.left t) i

= R (Tape1.left (b,l,r)) i

= R (match l with x::xs → (b,xs,x::r) | [ ] → (b,[],b::r)) i

=

Case analysis on l

= if i<1 then nth b l (-i) else nth b r (i-1)

= if i-1<0 then nth b l (-(i-1+1)) else nth b r (i-1)

= R (b,l,r) (i-1)

= Tape0.left (R (b,l,r)) i

= Tape0.left (R t) i

Solution to Problem 2c(left)

2c(read): $\forall t.\ R\ (Tape1.left\ t) = Tape0.left\ (R\ t)$

so we will prove: $\forall t.\ \forall i.\ R\ (Tape1.left\ t)\ i = Tape0.left\ (R\ t)\ i$

Suppose t=(b,l,r)

Case: l=[ ]

R (Tape1.left t) i

= R (Tape1.left (b,l,r)) i

= R (match l with x::xs → (b,xs,x::r) | [] → (b,[],b::r)) i

= R (b, [ ], b::r) i

= if i<0 then nth b [ ] (-(i+1)) else nth b (b::r) i

= if i<0 then b else if i=0 then b else nth b r (i-1)

= if i<0 then b else nth b r (i-1)

= if i<1 then nth b l (-i) else nth b r (i-1)

= if i-1<0 then nth b l (-(i-1+1)) else nth b r (i-1)

= R (b,l,r) (i-1)

= Tape0.left (R (b,l,r)) i

= Tape0.left (R t) i

Solution to Problem 2c(left)

2c(read): $\forall t.\ R\ (Tape1.left\ t) = Tape0.left\ (R\ t)$

so we will prove: $\forall t.\ \forall i.\ R\ (Tape1.left\ t)\ i = Tape0.left\ (R\ t)\ i$

Suppose t=(b,l,r)

Case: l=x::xs

R (Tape1.left t) i

= R (Tape1.left (b,l,r)) i

= R (match l with x::xs → (b,xs,x::r) | [ ] → (b,[],b::r)) I

= R (b,xs,x::r)

= if i<0 then nth b xs (-(i+1)) else nth b (x::r) i

= if i<0 then nth b xs (-(i+1)) else if i=0 then x else nth b r (i-1)

= if i<1 then nth b (x::xs) (-i) else nth b r (i-1)

= if i<1 then nth b l (-i) else nth b r (i-1)

= if i-1<0 then nth b l (-(i-1+1)) else nth b r (i-1)

= R (b,l,r) (i-1)

= Tape0.left (R (b,l,r)) i

= Tape0.left (R t) i

Done!

Warning: I did not show these as two-column proofs, which you will need to do in homeworks and exams. You can go back through the proofs and fill in the second column yourself, as an exercise.