Rules for Induction

1 plus (S(n) m) = S(n m)

(2) Length $|nill| = Z$

$|Cons(n; l)| = |l|+1$ or $S(l)$

(3) Concat = nill $\cdot l_2 = l_2$

$Cons(n; l_1) \cdot l_2 = Cons(n; l_1 \cdot l_2)$

(4) rev (nill) = nill

rev (Cons(n; nill)) = Cons(n; nill)

rev Cons(n; l) = rev l $\cdot$ Cons(n; nill)

or $\overset{+}{(+)}$ Concat (rev l; Cons (n; nill));

(5) Size (empty) = Z

~~Size (node(n; t_1; t_2))~~ = S (Size(t_1) + Size(t_2))

Size (node(n; t_1, t_2)) or Size(t_1) + Size(t_2) + 1

7) Height (empty) = empty

Height (node(n; t_1, t_2)) = S(max(H(t_1), H(t_2)))

-) LTR (Left to Right reversal)

LTR (empty) = empty

LTR (node(n; t_1, t_2)) = node (n; LTR(t_2), LTR(t_1))

(8) Preorder

pre (empty) = nill

pre (node(n; t_1, t_2)) = Cons (n; preorder(t_1)) $\cdot$ pre (t_2)

$|l_1 + l_2| = |l_1| + |l_2|$.

$S(n) \cdot m = (n+1)m = m \cdot m + m$

Inorder (empty) = nill
Inorder (n ; t_1 ; t_2) = Inoreder (t_1) · Cons (n ; Inorder (t_2))

Post order

l_1 · Cons (n ; nill)
= Cons (n ; l)

Post (empty) = nill
Post (n ; t_1 ; t_2) = Post (t_1) · Post (t_2) · Cons (n ; nill)

Post (t_1) · Cons (~~post~~ n ; Post (t_2))

Karan Rathore

11 0661622

Inductive type

nat = ind (t. unit + t).
list = ind (t. unit + nat × t)
tree = ind (t. unit + nat × t × t)

unit + nat = nat
unit + nat × list = list
unit + nat × tree × tree = tree

recursive term
rec {t.τ)(e₁; y.e₂):

$$\frac{\Gamma \vdash e : ind(t.\tau) \quad \Gamma, y : [\tau'/t]\tau \vdash e_1 : \tau'}{rec\{t.\tau\}(e_1; y.e_2) : \tau'}$$

Case: nat $\quad rec\{t.unit + t\}(e;\ \underbrace{y}_{type:\ unit + nat}.\underbrace{e_1}_{nat}) : nat$

(unit + nat) → nat // inl gives base case   unit → nat
Example: Double function   recursor // inr gives successor induction   nat → nat (Inductive)

d(z) = z

$d(s(n)) = s(s(\underbrace{d(n)}_{y}))$ nat — inductive of nat

rec {t. unit + t} (e; y.e₁) : nat

For d = lam{ind(t. unit + t)}(n. rec{t. unit + t}(n;
$y.case(y;\ \underbrace{u.z}_{\substack{Base \\ (unit \to nat)}};\ \underbrace{a.s(s(a))}_{successor\ (nat \to nat)}))$

Karan Rathore

110661622

example 2

$(t.\,unit + nat \times t)$

by rec len : list → nat

head of the list

len = lam {list} (l. lrec (l; z; h.y. s(y)))

Inductive case

len: ind(t. unit + nat × t) → ind(t. unit + t)

$\underbrace{\text{ind}(t.\,unit + nat \times t)}_{list} \to \underbrace{\text{ind}(t.\,unit + t)}_{nat}$

Base case

unit. t

len = lam {list} (l. rec {t. unit + nat×t} (l . y . case (y; u.z;
d . s(snd(d))))

y = unit + nat × nat

$\underbrace{y}_{nat}$ h y t = nat

because we want nat as a result

list → nat

$\underbrace{nat}_{head} \times \underbrace{nat}_{y\ (recursive\ call)}$

Successor case

Problem 3 sum(empty) = 0

node / leaf / root

Left Branch

sum(node(n; t₁; t₂)) = n + sum(t₁) + sum(t₂)

Right Branch

tree = ind(t. unit + nat × t × t)

unit + nat × (nat × nat)

Sum = lam {tree} (t. rec {t. unit + nat × t × t} (t; y . case (y; u.z; d.
fst(d) + fst(snd(d)) +
snd(fst(d)))))

d = nat × (nat × nat) = fst(d) + fst(snd(d)) + snd(snd(d))

$\underbrace{(nat \times nat)}_{snd(d)}$

$size(node(n; t_1, t_2))$ ✓✓

$Size(LTR(t)) = Size(t)$

By induction on t

Case: $t = empty$

$Size(LTR(empty))$

$Size(empty)$ ✓

Case: $t = node(n; t_1, t_2)$

$Size(LTR(node(n; t_1, t_2)))$

LTR Rule

$Size(\cancel{LTR}\ node(n; LTR(t_1), LTR(t_2)))$

$S(size(LTR(t_1)) + size(LTR(t_2)))$

By size Rule

By IH

$S(\cancel{LTR}\ size(t_1) + size(t_2))$

$size(node(n; t_1, t_2))$

$\ell \cdot c$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash let(e_1; x.e_2) : [redacted] \tau_2}$$

$(\exists x. (A(x) \to (B \times C))) \to (\forall x. A(x)) \to D$

$lam\{\exists x. (A(x) \to (B \times C))\}(x. \cancel{lam}\{\overset{A(x)\to D}{y}\}. lam\{A(x)\}(\overset{\tau A(x)}{z}. e))$ X

$e = ap(y; z)$;

$lam\{\exists x. (A(x) \to (B \times C))\}(x. lam\{\forall x. CA(x)\}(y.$

$$\frac{[e_2 \ val]}{ap(\underline{lam\{\tau\}}(x.e_1); e_2) \mapsto [e_2/x]e_1}$$

Preservation $\quad e : \tau$ & $\exists e, \ e \mapsto e'$ then $e' : \tau$

$e = ap(lam\{\tau\}(x.e_1); e_2) \quad : \tau_2$

$e' = [e_2/x]e_1 \quad : \boxed{\tau_2}$

By Inversion on $\boxed{e : \tau}$

$\Gamma \vdash \underbrace{lam\{\tau\}(x.e_1)}_{e_1} : arr(\underline{\tau_1} : \tau_2)$ — H_1

$\Gamma \vdash e_2 : \tau_1$ — H_2

By Inversion on H_1

$\Gamma, x : \tau_1 \vdash \underline{e_1} : \tau_2$ — H_3

By Substitution Theorem H_2 & H_3

$\Gamma, [e_2/x]e_1 : \tau_2$ ✓

$$\frac{\Gamma \vdash e_1 : num \quad \Gamma \vdash e_2 : num}{\Gamma \vdash plus(e_1 ; e_2) : num}$$

$\Delta, t\ type \vdash \tau\ type$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash let(e_1 ; x . e_2) : \tau_2}$$

$\Gamma \vdash e : \tau$, $\Gamma, x : \tau \vdash e' : \tau'$ then $[e/x]e' : \tau'$

$\Delta, t\ type \vdash \tau'\ type$ & $\Delta \vdash \tau\ type$ then $\Delta, \vdash [\tau/t] \vdash [\tau/t]\tau'\ type$

$\Delta, t\ type\ \Gamma \vdash e : \tau'$ & $\Delta \vdash \tau\ type$ $\quad \Delta, [\tau/t]\Gamma \vdash [\tau/t]e : \tau'$

pack$\{t.\tau\}\{t\}(e)$

open $\{t.$

$$\frac{\Delta\ \Gamma \vdash e_0 : \exists(t.\tau) \quad \Delta, t\ type\ \Gamma \vdash x : \tau \vdash e_1 : \tau_2 \quad \Delta \vdash \tau_2\ type}{open\{t.\tau\}\{\tau_2\}(e_0; t, x . e_1) : \tau_2}$$

$$\frac{\Delta \vdash t\ type \vdash \tau\ type \quad \Delta \vdash \rho\ type \quad \Delta\ \Gamma \vdash e : [\rho/t]\tau}{pack\{t.\tau\}\{\rho\}(e) : \exists(t.\tau)}$$

$$\frac{\Delta\ \Gamma \vdash e : \forall(t.\tau') \quad \Delta \vdash \tau\ type}{\Delta\ \Gamma \vdash App\{\tau\}(e) : [\tau/t]\tau'}$$

Karan Rathore

110661622

$$\frac{e_1 \to e_1'}{join(e_1; e_2) \to join(e_1'; e_2)} - R_1$$

$$\frac{e_1\ val \quad e_2 \to e_2'}{join(e_1; e_2) \to join(e_1; e_2')} - R_2$$

Case 1 $\quad e_1 \mapsto e_1'$

$join(e_1; e_2) \mapsto join(e_1'; e_2)$ By R_1

Take $e' = join(e_1'; e_2)$ Then $e \mapsto e'$

Case 2 $\quad e_1$ val

By IH on H2 $\quad (\Gamma \vdash e_2 : \tau_2)$

e_2 val or $\exists e_2' ;\ e_2 \mapsto e_2'$

Case 2.1 $\quad e_2 \mapsto e_2'$ By Rule R_2

$join(e_1; e_2) \mapsto join(e_1; e_2')$

Take $e' = join(e_1; e_2')$ Then $e \mapsto e'$

Cas 2.2 $\quad e_2$ Val

~~[illegible]~~

By Rule R2

e_1 val

Similary, e_2 Val also

$e = join(e_1; e_2)$ value

e is value

Karan Rathore

110661622

Inductive Type

z , $S(n)$ $\quad$ $S: nat \to nat$

z $\quad$ $S(z)$ $\quad$ $S(S(z))$ .....

0 $\quad$ 1 $\quad$ 2

list:

nil, cons(x; l) : $\quad$ cons: $\tau \times list(\tau) \to list(\tau)$

nil; cons(z; nil); cons(S(z); cons(z; nil)) ....

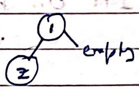
[] $\quad$ [0] $\quad$ [1, 0]

tree :

left tree $\quad$ Right tree

empty, node(x; t_1; t_2) $\quad$ node: $\tau \times tree \times tree \to tree$

empty; node(z; empty; empty); node(S(z); node(z; empty; empty); empty)



Constructor	Destructor
"Produce"	"Use"

rec(e; e_0; x.y.e_1) $\quad$ (z, rec, S)

listrec(e; e_0; h.t.y.e_1) $\quad$ (list(τ), cons, nil)

trec(e; e_0; n.l.r.y.e_1) $\quad$ (tree(τ), z, empty)

nat $\quad$ $(unit + nat) \to nat = unit \to nat \times nat \to nat$

list(nat) $\quad$ $(unit + (nat \times list(nat)) \to list(nat)$

tree $\quad$ $(unit + (nat \times tree \times tree)) \to tree$

leq z m = s(z)
leq s(n) z = z
leq s(n) s(m) = leq n m

$\overset{n}{nat} \to \overset{m}{nat} \to nat$

lam{nat}(n.lam{nat}(m.rec{s(z); $\underset{n}{x}$.$\underset{leq\ n}{y}$.rec{z; $\overset{m}{p}$.q.y(p)}(m)}(n)))

sub z m = z
sub s(n) z = s(n)
sub s(n) s(m) = sub n m

$nat \to nat \to nat$

lam{nat}(n.lam{nat}(m.rec{z; $\underset{n}{x}$.$\underset{sub\ n}{y}$.rec{s(x); $\underset{m}{x'}$.y'.y(x')}(m)}(n)))

fib z = z
fib s(z) = s(z)
~~fib~~ fib s(n) = fib(n) + fib(n+1)

$nat \to nat$

lam{nat}(n.rec{z; $\overset{z}{x}$.$\overset{fib\ s(z)}{y}$.rec{s(z); p.q.plus(q.$\underset{n}{y(p)}$; $\underset{n+1}{s(y(p))}$)}(n)})

f n 0 = n
f n $\underbrace{(m+1)}_{s(m)}$ = m × f(n m) + 1

$\overset{n}{nat} \to \overset{m}{nat} \to nat$

lam{nat}(n.lam{nat}(m.rec{n; $\overset{m}{x}$.y.s(mult(x;y))}

10) TSUB n Z = n

TSUB n $S(m)$ = pred (TSUB n m)

nat → nat → nat

lam{nat}(n: lam{nat}(m. rec{n; x.y. pred(y)}(m)))

11) lt Z Z = Z

lt Z S(m) = S(Z)

lt S(n) Z = Z

lt $S(n_1)$ $S(n_2)$ = lt n_1 n_2

nat → nat → nat

lam{nat}(n: rec{lam{nat}(m. rec{Z; x.y. S(Z)}(m)); p.q. rec{Z; u.v. q(u)}(n_2)}(n_1))

lam{nat}(n. rec{lam{nat}(m). rec{Z; x.y. S(Z)}(m); p.q. rec{Z; u.v. q(u)}(n_2)}(n_1))

12) gt 0 0 = 0

gt S(m) 0 = S(Z)

gt 0 $S(n_1)$ = 0

gt $S(n_2)$ $S(n_1)$ = gt n_2 n_1

nat → nat → nat

lam{nat}(n. rec{lam{nat}(m. rec{Z; x.y. S(Z)}(m); p.q. rec{Z; u.v. q(u)}(n_2)}(n_1))

Let Rule

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma, x:\tau \vdash e_2 : \tau'}{\Gamma \vdash let(e_1; x.e_2) : \tau'}$$

$$\frac{e_1 \mapsto e_1'}{let(e_1; x.e_2) \mapsto let(e_1'; x.e_2)}$$

$$\frac{\cancel{e_2\ val\ e_2 \mapsto e_2'}}{let(e_1; x.e_2) \mapsto [e_1/x]e_2}$$

$$\frac{e_1\ val \quad e_2 \mapsto e_2'}{let(e_1; x.e_2) \mapsto let(e_1; x.e_2)}$$

System T

$$\frac{\Gamma \vdash e : nat}{\Gamma \vdash s(e) : nat} \qquad \frac{}{\Gamma \vdash z : nat}$$

$$\frac{\Gamma \vdash e_1 : \to(\tau_2; \tau) \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash ap(e_1; e_2) : \tau_2}$$

$$\frac{\Gamma \vdash e : nat \quad \Gamma \vdash e_0 : \tau \quad \Gamma, x : nat, y : \tau \vdash e_1 : \tau}{\Gamma \vdash rec\{e_0; x.y.e_1\}(e) : \tau}$$

Pros

$$\frac{\Gamma \vdash x : \tau_1 : e : \tau_2}{\Gamma \vdash lam\{\tau\}(x.e) : arr(\tau_1; \tau_2)}$$

$$\frac{e_1 \mapsto e_1'}{ap(e_1; e_2) \mapsto ap(e_1'; e_2)} \qquad \frac{e_1\ val \quad e_2 \mapsto e_2'}{ap(e_1; e_2) \mapsto ap(e_1; e_2')}$$

$$\frac{e_2\ val}{\cancel{ap(e_1; e_2)}\ ap(lam\{\tau\}(x.e_1); e_2) \mapsto [e_2/x]e_1}$$

$$\frac{e \mapsto e'}{rec\{e_0; x.y.e_1\}(e) \mapsto rec\{e_0; x.y.e_1\}(e')}$$

$$\frac{}{lam\{\tau\}(x.e)\ val} \qquad \frac{}{rec\{e_0; x.y.e_1\}(z) \mapsto e_0}$$

$$\frac{}{rec\{e_0; x.y.e_1\}\ s(e) \mapsto [e, rec\{e_0; x.y.e_1\}(e)/x; y]e_1}$$

$$[e/x][rec\{e_0; x.y.e_1\}(e)/y]e_1$$

app Progress

$$\frac{\Gamma \vdash e_1 : arr(\tau_2;\tau_1) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash app(e_1;e_2) : \tau_2}$$

$e : \tau$ then either e val or $\exists e', e \mapsto e'$

$e = app(e_1;e_2)$

$\tau : \tau_2$

By Above the line

$\Gamma \vdash e_1 : arr(\tau_2;\tau_1)$ — H_1

$\Gamma \vdash e_2 : \tau_2$ — H_2

By IH on H_1

$e_1 \mapsto e_1'$ or e_1 val

case $e_1 \mapsto e_1'$

take e_1' as $ap(e_1';e_2)$

$e_1 \mapsto e_1'$ By rule $\dfrac{e_1 \mapsto e_1'}{ap(e_1;e_2) \mapsto ap(e_1';e_2)}$

case e_1 val

By IH on H_2

$e_2 \mapsto e_2$ or e_2 val

e_2 val

e_1 val & $e_1 : arr(\tau_2;\tau_2)$ then By CFL

$e_1 = lam\{\tau\}(x.e)$

By Inversion on e_1

$\Gamma, x : \tau_2 ; \tau_1$ — H_3

By Substitution Theorem

$[e_2/x]e_1$ By rule $\dfrac{[e_2 \text{ val}]}{ap(lam\{\tau\}(x.e);e_2) \mapsto [e_2/x]e_1}$

e_2 val

$e_2 \mapsto e_2'$

take e_2' as $ap(e_1;e_2')$

$e_2 \mapsto e_2'$

$$\frac{}{rec\{e_0; x.y.e_1\}(z) \mapsto e_0}$$ val P_n

$e = rec\{e_0; x.y.e_1\}(z)$ $\quad e : \tau$

$e' = e_0$

By Inversion on $e : \tau$

$\Gamma \vdash e_0 : \tau$ - H_1

$\Gamma \vdash e : nat$ - H_2

$\Gamma, x : nat, y : \tau \vdash e_1 : \tau$ - H_3

$\boxed{\text{Goal } e_0 : \tau}$

$$\frac{\Gamma \vdash e_0 : \tau \quad \Gamma \vdash e : nat \quad \Gamma, x : nat, y : \tau \vdash e_1 : \tau}{\Gamma \vdash rec\{e_0; x.y.e_1\}(e) : \tau}$$

Progress $e : \tau$ then e is val or $\exists e, e \mapsto e'$

Case $e \mapsto e'$

take e' g $rec\{e_0; x.y.e_1\}(e)$

then By rec rule $\boxed{e \mapsto e'}$

Case e val

Above the line

$\Gamma \vdash e_0 : \tau$

$\Gamma \vdash e : nat$

$\Gamma, x : nat, y : \tau \vdash e_1 : \tau$

If $e : nat$ & e val then $e = z$

$e = s(e)$

Case $e = z$ take e' as e_0

By rec rule $\frac{}{rec\{e_0; x.y.e_1\}(z) \mapsto e_0}$

e val

Case $e = s(e)$

take $e' = [e, rec\{e_0; x.y.e_1\}e/x, y]$

$$\frac{s(e) \text{ val}}{rec\{e_0; x.y.e_1\}(s(e)) \mapsto}$$

$\boxed{s(e) \text{ val}}$

Preservation

$$\frac{S(e)\ Val}{rec\{e_0; x, y.e_1\}(S(e)) \mapsto [e, rec\{e_0; x.y.e_1\}(e)/x, y]e_1}$$

$e = rec\{e_0; x.y.e_1\}(S(e))$
$e' = [e, rec\{e_0; x.y.e_1\}(e)/x, y]e_1$

Above the line
$S(e)$ Val — H_0

By Inversion on $e:\tau$ — $\Gamma \vdash e_0 : \tau$ — H_1
$\Gamma \vdash S(e) : nat$ — H_2
~~$\Gamma \vdash e_1 : \tau$~~ — H_3
$\Gamma, x : nat, y : \tau \vdash e_1 : \tau$ — H_4

By Inversion on H_2
$\Gamma \vdash e : nat$ — H_5

By Apply weakening on H_3
$\Gamma, y : \tau \vdash e_0 : \cancel{\tau}\, nat$ H_6

By Substitution on H_4 & H_6

$\Gamma, y : \tau \vdash [e/x]e_1$ H_7

By rule H_1, H_3, H_5

$\Gamma \vdash rec\{e_0; x.y.e_1\}(e) : \tau$ {By rec rule} — H_8

By substituting H_8 in H_7

$\Gamma, [e/x, [rec\{e_0; x.y.e_1\}(e)/y]]e_1 : \tau$
$e' : \tau$

$$\frac{\Gamma \vdash e : nat}{\Gamma \vdash S(e) : nat}$$

$$\frac{e \mapsto e'}{S(e) \mapsto S(e')}$$

$e : S(e)$
$e' : S(e')$
Above the line
$e \mapsto e'$

By Inversion on $e:\tau$
$\Gamma \vdash e : nat$ — H_1

By IH on H_1
$e' : nat$
By rule $S(e') : nat$

$$\frac{e\ Val}{S(e)\ Val}$$