به نام خدا

محمدمهدی آقاجانی

۹۳۳۱۰۵۶

تمرین دوم

استاد : دکتر سلیمان فلاح

فصل سوم

۳.۵.۱۰.

$t \to t'$ ------- $t \to^* t'$

$t \to t'$ and $t' \to^* t''$ ------ $t \to^* t''$

۳.۵.۱۷.

$v \downarrow v = v \to v$

$t1 \downarrow true\ t2 \downarrow v2$ ---------- $if\ t1\ then\ t2\ else\ t3 \downarrow v2$

$= t1 \to true\ t2 \to v2$ ---------- $If\ t1\ then\ t2\ else\ t3 \to if\ true\ t2\ else\ t3 \to t2 \to v2$

$t1 \downarrow false\ t3 \downarrow v3$ ---------- $if\ t1\ then\ t2\ else\ t3 \downarrow v3$

$= t1 \to false\ t3 \to v3$ ---------- $If\ t1\ then\ t2\ else\ t3 \to if\ fasle\ t2\ else\ t3 \to t3 \to v3$

$t1 \downarrow nv1$ ---------- $succ\ t1 \downarrow succ\ nv1$

$= t1 \to nv1$ ---------- $succ\ t1 \to succ\ nv1$

t1 ↓ 0 = t1 → 0

---

pred t1 ↓ 0 pred t1 → pred 0 0

t1 ↓ succ nv1 = t1 → succ nv1

---

pred t1 ↓ nv1 pred t1 → pred succ nv1 → nv1

t1 ↓ 0 = t1 → 0

---

iszero t1 ↓ true iszero t1 iszero 0 → true

t1 ↓ succ nv1 = t1 → succ nv1

---

iszero t1 ↓ false iszero t1 → iszero succ nv1→ false

مشخص است که اثبات t ↓ v if t →* v مانند موارد بالاست

۳.۵.۱۸.

t2 and t3 are terminal values

If true then t2 else t3 → t2

t2 and t3 are terminal values

If false then t2 else t3 → t3

t2 and t3 are terminal values and t1 → t1’

If t1 then t2 else t3 → If t1’ then t2 else t3

t2 → t2’ and t3 → t3’

---

if t1 then t2 else t3 → if t1 then t2’ else t3’

آخرین مطمئن میشود که بدنه then/else زودتر از t1 پردازش میشود و بقیه قوانین برای اطمنیان از این هستند که if بیرونی زودتر پردازش می‌شوند.

فصل پنجم

۵.۲.۲

$succ$ = λn.λs.λz s(n s z)

$plus$ 1 n = λn.λs.λz (λs.λz (s z)) s (n s z) = λn.λs.λz s(n s z) = succ n

۵.۲.۳.

mul = λx.λy.λs.λz.x(y s)z

5.2.4

λm.λn (m (Times n) c1) is equivalent to n^m5.2.7

equal = λx . λy . and (iszero (x prd y)) (iszero (y prd x))

۵.۲.۱۰

f = λx . λy . if iszero x then c0 else scc (y (pred x))

churchnat = fix f

۵.۲.۱۱.

sum = λm . λn . test(isnil n)(λx.c0)(λx.(plus(head n)(m(tail n))))c0

sumlist = fix sum

فصل چهارم

۴.۴

a)

((λf.λg. f (g 1)) (λx.x+4)) (λy.3-y) -> (λg.(λx.x+4) (g 1)) (λy.3-y) ->

(λg. g 1+4) (λy.3-y) -> (λy.3-y) 1+4 -> 3-1+4

b)

((λf.λg. f (g 1)) (λx.x+4)) (λy.3-y) -> (λg.(λx.x+4) (g 1)) (λy.3-y) ->

(λx.x+4) ((λy.3-y) 1) -> (λy.3-y) 1+4 -> 3-1+4

۴.۵.

(λcompose.(λh.compose h h 3)(λx.x+x))(λf.λg.λx. f(g x)) →

(λh.(λf.λg.λx.f(g x))h h 3)(λx.x+x) →

(λf.λg.λx.f(g x))(λx.x+x)(λx.x+x)3 →

(λg.λx.(λx.x+x)(g x))(λx.x+x)3 →

(λx.(λx.x+x)((λx.x+x)x))3 →

(λx.(λx.x+x)(x + x))3 →

(λx.((x+x)+(x + x)))3 →

((3+3)+(3+3)) → 12

در خط سوم از این کاهش عبارت (λf.λg.λx.f(g x))(λx.x+x)(λx.x+x)3 را داریم (که با جایگزینی (λx.x+x) به جای h در عبارت compose h h 3 این کاهش بسیار سریعتر شد) که بدین معناست که x ابتدا بر g، apply میشود و سپس نتیجه بر f، apply می‌شود.

۴.۶

f = λf.ff

$f(f)$: (λf.ff)(λg.gg) → (λg.gg)(λg.gg) → (λg.gg)(λg.gg)

این تابع به ترم نمیرسد و دایما تکرار میشود

۴.۸.

a) $C[[x:=1;x:=x+1;]](s0)=C[[x:=x+1]](C[[x:=1]](s0))=C[[x:=x+1]](s1)=s2$

s0 = {(x,0)}

s1 = {(x,1)}

s2 = {(x,2)}

b)

چون در این عبارت پس از این دو تغییر تنها مقدار x (از یک) یک واحد افزایش می‌یابد و هیچ متغیر دیگری افزایش نمی‌یابد، می‌توان این تغییر را در یک مرحله نشان داد و stat را فقط modify کرد و چون مقدار x در s قرارداده شده است پس مقدار x در s در محاسبات تاثیرگذار نیست.

s = {(x,uninit)}

s1 = {(x,1)}

s2 = {(x,2)}

۴.۹

a)

C[[x:=1;x:=x+1]](s)=C[[x:=x+1]](C[[x:=1]](s))=C[[x:=x+1]](s1)=s2

$C[[x := 0; y := 0;$ if $x = y$ then $z := 0$ else $w := 1$ $]](s0)$ =

$C[[\ y := 0;$ if $x = y$ then $z := 0$ else $w := 1$ $]](C[[x := 0;]](s0))$

$C[[x := 0;]](s0)=modify(s0,x,0)=s1 \rightarrow$

$C[[\ y := 0;$ if $x = y$ then $z := 0$ else $w := 1$ $]](s1)$ =

$C[[$ if $x = y$ then $z := 0$ else $w := 1$ $]](C[[y := 0;]](s1))$

$C[[y := 0;]](s1)=modify(s1,y,0)=s2 \rightarrow$

$C[[$ if $x = y$ then $z := 0$ else $w := 1$ $]](s2) = if\ E[[\ x == y]](s2)\ then$

$C[[z := 0]](s2)\ else\ C[[w := 1]](s2) = if\ E[[\ x == y]](s2)\ then\ modify(s2,z,0) else\ modify(s2,w,1) = if\ E[[\ 0 == 0]](s2)\ then\ modify(s2,z,0) else\ modify(s2,w,1) = modify(s2,z,0)$

b)

C[[if $x = y$ then z := y else z := w]](s0) = $if\ E$[[$x == y$]](s0) $then\ C$[[$z := y$]](s0) $else\ C$[[$z := w$]](s0) = if E[[$x == y$]](s0) $then\ modify$(s0,z,y)$else\ modify$(s0,z,w)

۴.۱۱

a) بله وقتی که زمان ارزیابی e1 طولانی باشد و g منتظر ارزیابی "if(e1=0)"بماند

B) در ارزیابی g مقدار return 1 میشود

C) میتوانیم e1 , e2 را موازی ارزیابی کنیم و بعد g را ارزیابی کنیم

d) خیر . در این شرایط نمیتوان g را موازی ارزیابی کرد زیرا در ارزیابی موازی نتیجه میتواند مقادیر متفاوتی داشته باشد

۴.۱۳

A) به دلیل وجود انتساب و داشتن side effect نمیتوان ارزیابی موازی داشت به دلیل ویژگی ذاتی زبان های functional میتوانند در سیستم های موازی و همروند به کار بروند. اما از طرفی به خاطر اینکه بیشتر سیستم های قبلی با زبان های imperative توسعه داده شده اند این زبان ها بیشتر در بین مردم رایج هستند

b) زبان های imperative در زمان اجرا به منابع کمتری نیاز دارند ولی زبان های functional در واقع سربار بیشتری دارند در نتیجه منابع بیشتری استفاده میکنند

c) زبان های functional به دلیل عملیات بیشتر حین اجرا از جمله garbage collection به فایل های اجرایی بزرگتری تبدیل میشوند

D) خوانایی و قابل اعتماد بودن و راحتی امروزه در بسیاری از کاربرد ها مهم تر از سرعت می باشد لذا زبان های imperative محبوبیت بیشتری دارند اما زبان های functional به دلیل پتانسیل بالاتر در پردازش موازی ، در این کاربرد ها بیشتر استفاده میشوند

e) امروزه با افزایش حجم حافظه ها و پیشرفت سخت افزار دیگر نگرانی های قبلی وجود ندارد البته هنوز استفاده بهینه مطرح میباشد.

۴.۱۴.

۱) بعضی از زبانهای functional برای همروندی و موازی سازی طراحی نشدند و لذا به طور کلی این قابلیت را ندارند.

۲) ممکن است بعضی از زبانهای functional در ظاهر syntax ، expression-based باشند ولی در هنگام implementation، طبق زبانی imperative شود(از assignment ها برای ایجاد خاصیت استفاده شده)

۳) درست است که زبان های functional راهی را برای اجرای برنامه های parallel فراهم می کنند اما باید توجه کرد که ما نیاز به دسترسی به تغییراتی که ایجاد شده داریم که باتوجه به اینکه هیچ اثری از خود به جا نمیگذارند و state خاصی ندارند . در نتیجه اگر بخواهیم نتیجه ی چندکار موازی را بهم ترکیب کنیم در زبان های functional امکان پذیر نیست.مشکل دیگر این است که امکان کنترل کردن ترتیب اجرا در قسمت های موازی در زبان های pure functional وجود ندارد

۴) اگر تعداد ترد ها (سطح موازی سازی) زیاد باشد سربار زیادی دارد(زمان context switch خیلی زیاد می شود و به صرفه نیست)

۵) زبان های functional در استفاده از ساختار داده هایی مانند آرایه های یک بعدی بهینه نیستند و در صورتی که برنامه از این ساختار داده ها زیاد استفاده کند عملکرد خوبی نخواهد داشت.

۶) یکی از مشکلات زمانی رخ می دهد که دو نخ بخواهند هم زمان به یک critical section وارد شوند و گاهی نیز ممکن است stateهای مشترک پیش بیایند .

۷) یکی دیگر از مشکلات این زبان ها آن است که خروجیشان به اندازه ی دیگر زبان ها به دلیل عدم تطابق هایی در برخی دستورات بین زبان برنامه که functional است و زبان ماشین که imperative است، بهینه نیست .