به نام خدا

شیرین حاجی امین شیرازی ۹۱۳۱۰۰۹ **تمرین۳**

تمرینات کتاب concepts in programming languages

۳-

$$(\lambda\, x.\lambda y\, .\, xy\,)(\, \lambda x\, .\, xy) \rightarrow\ \lambda y\, .\, (\lambda x\, .\, xy)y \rightarrow\ \lambda y\, .\, yy$$

$$(\lambda x\, .\, \lambda y\, .\, xy)(\lambda x.\, xy\,) \rightarrow (\lambda x.\, \lambda y.\, xy)(\lambda m.\, mn) \rightarrow \lambda y.\, (\lambda m.\, mn)y \rightarrow \lambda y.\, yn$$

در صورت عدم نام گذاری مشاهده می شود که متغییر bound شده دوبار در خروجی ظاهر می شود در حالی که اگر نام گذاری صورت گیرد ، متغییر آزاد n نیز مشاهده می شود.

۴- (a

$$\Big((\lambda f.\, \lambda g.\, f(g\ 1))(\lambda x.\, x+4)\Big)\, (\lambda y.\, 3-y) \rightarrow\ \big(\lambda g.\, (\lambda x.\, x+4)(g\ 1)\big)(\lambda y.\, 3-y) \rightarrow (\lambda x.\, x+4)\big((\lambda y.\, 3-y\,)1\big) \rightarrow \big((\lambda y.\, 3-y)1\big)+4 \rightarrow 3-1+4 \rightarrow 6$$

(b

$$\Big((\lambda f.\, \lambda g.\, f(g\ 1))(\lambda x.\, x+4)\Big)\, (\lambda y.\, 3-y) \rightarrow\ \big(\lambda g.\, (\lambda x.\, x+4)(g\ 1)\big)(\lambda y.\, 3-y) \rightarrow (\lambda x.\, x+4)\big((\lambda y.\, 3-y\,)1\big) \rightarrow (\lambda x.\, x+4)(3-1) \rightarrow 2+4 \rightarrow 6$$

۶-

$$f\ = \lambda g.\, gg$$

$$f(f) = (\lambda g.\, gg)(\lambda g.\, gg) \rightarrow (\lambda g.\, gg)(\lambda g.\, gg) \rightarrow (\lambda g\ .\, gg)(\lambda g.\, gg) \rightarrow \cdots$$

این فراخوانی تابع هیچ گاه تمام نمی شود و این عبارت به مقدار دیگری کاهش نمی یابد و تا ابد ادامه می یابد .

۸-(a

$$C\big[[x \coloneqq 1; x \coloneqq x+1;]\big](s_0) = C[[\ x \coloneqq x+1]](C\big[[x \coloneqq 1]\big](s_0)) = C\big[[x \coloneqq x+1]\big](modify\ (s_0, x, \big[[1]\big]\big) = modify(s_1, x, 2)$$

(b

چون در عبارت فوق، همیشه پس از این دو تغییر ، مقدار x از یک ، یک واحد افزایش می یابد ، و هیچ متغییر دیگری تغییر نمی یابد ، می توان این تغییر را در یک مرحله نشان داد و stat را فقط modify کرد .

$$C[[x \coloneqq 1; x \coloneqq x+1;]](s) = C[[x \coloneqq 2]](s) = modify\ (s, x, 2)$$

۹–a)

$$C[[y \coloneqq 0; if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]]\left(C[[x \coloneqq 0]](s_0)\right)$$

$$= C[[y \coloneqq 0; if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]] modify(s_0, x, 0)$$

$$= C[[y \coloneqq 0; if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]](s_1)$$

$$= C[[if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]]\left(C[[y \coloneqq 0]](s_1)\right)$$

$$= C[[if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]] modify(s_1, y, 0)$$

$$= C[[if\ x = y\ then\ z \coloneqq 0\ else\ w \coloneqq 1]](s_2)$$

$$= if\ E[[x = y]](s_2) = err\ or\ C[[z \coloneqq 0]](s_2) = error\ or\ C[[w \coloneqq 1]](s_2) = error\ then\ error$$

$$else\ C[[z \coloneqq 0]](s_2) = modify\ (s_2, z, 0)$$

چون در هر دو قسمت If، z و w مقدار نمیگیرند، uninitialized باقی می مانند . (برخورد محافظه کارانه)

b)

$$C[[if\ x = y\ then\ z \coloneqq y\ else\ z \coloneqq w]](s)$$

$$= if\ E[[x = y]](s) = err\ or\ C[[z \coloneqq y]](s) = error\ or\ C[[z \coloneqq w]](s) = error\ then\ error$$

$$else\ if\ E[[x = y]](s) = true\ then\ C[[z \coloneqq y]](s)\ else\ C[[z \coloneqq w]](s)$$

بازهم در اینجا هر دو z وw ، uninit هستند زیرا در دو قسمت if ، هر دویشان init نشده اند.

۱۳–

a. از آنجایی که دستورات زبان های Imperative ساده ترند و به زبان ماشین نزدیک ترند ، تبدیل آنها به زبان ماشین بهینه تر است . هرچند در مورد پردازش های موازی ممکن است زبان های functional بهتر باشند.
b. چون برنامه های Imperative ساده ترند احتمالا فضای کمتری را برای محاسبات نیاز دارند.
c. به همان دلیل گفته شده در a که دستورات imperative به زبان ماشین نزدیکترند ، functional ها احتمالا ا دستورات پیچیده تری استفاده می کنند و بهینه نیستند و در نتیجه فایل های اجرایی بزرگتری دارند.
d. کمتر از زبانهای functional استفاده شد .
e. چون محدودیت حافظه امروز مطرح نیست این مسائل تا حدی حل شده اند.

-۱۴

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

تمرینات کتاب types and programming languages

-۲-۲-۵

$$successor(c_i) = \lambda n.\lambda s.\lambda z\ ns(sz)c_i \to \lambda s.\lambda z\ c_i s(sz) \to \lambda s.\lambda z.(\lambda s\ \lambda z.\underbrace{s(s(\ldots s(sz))}_{n\ \text{بار}}))\ \ s(sz)$$

$$\to \lambda s.\lambda z\underbrace{s(s(\ldots s(sz))}_{n+1\ \text{بار}} = \boldsymbol{c_{i+1}}$$

-۴-۲-۵

$$\mathrm{POW} := \lambda b.\lambda e.e\ b$$

$$\equiv \lambda m.\lambda n\ m\ (times\ n)c_1 \quad = n^m$$

-۸-۲-۵

$$\mathrm{nil} \coloneqq \lambda x.\mathrm{TRUE} \quad \equiv \mathrm{pair\ true\ true}$$

$$isnil \coloneqq first$$

$$head \coloneqq \lambda z.first(second\ z)$$

$$tail \coloneqq \lambda z.second\ (second\ z)$$

$$cons \coloneqq \lambda h.\lambda t.pair\ false(pair\ h\ t)$$

-۱۱-۲-۵

Call by name fixed point combinator : $y = \lambda h.\big(\lambda x.h(xx)\big)\big(\lambda x.h(xx)\big)$

$$sum = \lambda f\ \lambda list\ if\ isnill(list) then\ c_0 else\ plus\ head(list)\ sum\big(tail(list)\big)$$

حال برای تعریف تابع مورد نظرمان باید *fixed point* تابع *sum* را به دست آوریم: $y\ sum$

-۶-۳-۵

Full beta reduction:

$\frac{t_1 \to t_1'}{t_1 t_2 \to t_1' t_2}$

$\frac{t_2 \to t_2'}{t t_2 \to t t_2'}$

$(\lambda x.t)t_1 \to [x \to t_1]t$

normal form :

مثل حالت قبل ، فقط سمت چپ ترین قبل از همه کاهش می یابد

$\frac{t_1 \to t_1'}{t_1 t_2 \to t_1' t_2}$

$\frac{t_2 \to t_2'}{t t_2 \to t t_2'}$ (t must not be a reducible redex!:))

$(\lambda x.t)t_1 \to [x \to t_1]t$

$\lambda x.t \to \lambda x.t'$

Lazy:

در این روش ، *abstract reduction* مجاز نیست

$\frac{t_1 \to t_1'}{t_1 t_2 \to t_1' t_2}$

$(\lambda x.t)t_1 \to [x \to t_1]t$