Chris Fenton
CNC - formallang
Assignment 4

Chapter 9

1. Construct Turing machines with input alphabet {a, b} that compute the specified functions. The symbols u and v represent arbitrary strings over {a, b}*.

(a) f(u) = aaa

```
; <current state> <current symbol> <new symbol> <direction> <new state>

0   * * * era
era a _ r era
era b _ r era
era _ a r aa*
aa* _ a r aaa
aaa _ a l gtb
gtb a a l gtb
gtb _ _ r halt-accept
```

(b) f(u) = if length(u) is even then a; else b

```
; <current state> <current symbol> <new symbol> <direction> <new state>

; initial state
0 * * * odd

; odd case (odd)
odd a _ r oge
odd b _ r oge
odd A a * halt-accept

; odd - goto end (oge)
oge a * r oge
oge b * r oge
oge A B l ogh
oge _ B l ogh

; odd - goto home
ogh a * l ogh
ogh b * l ogh
ogh _ * r eve
```

```
; even case (eve)
eve a _ r ege
eve b _ r ege
eve B b * halt-accept

; even - goto end (ege)
ege a * r ege
ege b * r ege
ege B A l egh

; even - goto home
egh a * l egh
egh b * l egh
egh _ * r odd
```

(c) $f(u) = u^R$

```
; <current state> <current symbol> <new symbol> <direction> <new state>

; initial state
0   * * * gte

; goto end (gte)
gte a * r gte
gte b * r gte
gte _ * l cpy

; copy (cpy)
cpy a x r psa
cpy b x r psb

; paste a (psa)
psa _ a l gth
psa a * r psa
psa b * r psa
psa x * r psa

; paste b (psb)
psb _ b l gth
psb a * r psb
psb b * r psb
psb x * r psb

; goto home (gth)
gth a * l gth
gth b * l gth
gth x * l gth
```

```
gth _ * r cfd

; check for done (cfd)
cfd x * * cln
cfd a * * fch
cfd b * * fch

; find char (fch)
fch x * l cpy
fch a * r fch
fch b * r fch

; cleanup (cln)
cln x _ r cln
cln a * * halt-accept
cln b * * halt-accept
```

(d) f(u, v) = if length(u) > length(v) then u; else v

```
; <current state> <current symbol> <new symbol> <direction> <new state>

0   * * * gte

; goto end and mark
gte a * r gte
gte b * r gte
gte _ * r che
che a * r gte
che b * r gte
che _ * l sep
sep _ | l gth

; goto home (gth)
gth a * l gth
gth b * l gth
gth A * l gth
gth B * l gth
gth $ * l gth
gth _ * l chh
chh a * l gth
chh b * l gth
chh A * l chh
chh B * l chh
chh _ * r shr
shr * * r cpy

; copy initial input (cpy)
```

```
cpy a A r psa
cpy A * r cpy
cpy b B r psb
cpy B * r cpy
cpy $ * r cpy
cpy _ $ r pss
cpy | * r len

; paste a (psa)
psa a * r psa
psa b * r psa
psa _ * r psa
psa | * r paa
paa _ a l mls
paa a * r paa
paa b * r paa
paa $ * r paa

; paste b (psb)
psb a * r psb
psb b * r psb
psb _ * r psb
psb | * r pbb
pbb _ b l mls
pbb a * r pbb
pbb b * r pbb
pbb $ * r pbb

; paste _ (pss)
pss a * r pss
pss b * r pss
pss | * r ps*
ps* _ $ l mls
ps* a * r ps*
ps* b * r ps*

; move left to sep (mls)
mls a * l mls
mls b * l mls
mls $ * l mls
mls | * l gth

; length (len)
len a * * mk1
len b * * mk1
len x * r len
len $ * r ck2
```

```
; mark 1st word (mk1)
mk1 a x r gt2
mk1 b x r gt2
mk1 $ * r ck2

; goto 2nd word (gt2)
gt2 a * r gt2
gt2 b * r gt2
gt2 $ * r mk2

; mark 2nd word (mk2)
mk2 a x l gts
mk2 b x l gts
mk2 x * r mk2
mk2 _ * * gr1 ;1st word is longer

; goto sep (gts)
gts x * l gts
gts $ * l gts
gts a * l gts
gts b * l gts
gts | * r len

; check 2nd word for done (ck2)
ck2 a * * gr2 ;2nd word is longer
ck2 b * * gr2 ;2nd word is longer
ck2 _ * * gr2 ;2nd word is equal len
ck2 x * r ck2

; greater 2nd word (gr2)
gr2 * * * cl2

; greater 1st word (gr1)
gr1 * * * cl1

; cleanup 1st word (cl1)
cl1 $ _ l cl1
cl1 a _ l cl1
cl1 b _ l cl1
cl1 x _ l cl1
cl1 _ _ l cl1
cl1 | _ l er2
er2 A _ l er2
er2 B _ l er2
er2 $ _ l mfe
mfe A a l mfe
mfe B b l mfe
mfe _ * r halt
```

```
; cleanup 2nd word (cl2)
cl2 $ _ l cl2
cl2 a _ l cl2
cl2 b _ l cl2
cl2 x _ l cl2
cl2 _ _ l cl2
cl2 | _ l cv2
cv2 A a l cv2
cv2 B b l cv2
cv2 $ _ l er1
er1 A _ l er1
er1 B _ l er1
er1 _ * * mfs
mfs _ * r mfs
mfs a * * halt
mfs b * * halt
```

2. Let M = (Q, Σ, Γ, ▫, q_0, q_f) be a Turing machine that computes the partial characteristic function of the language L. Use M to build a standard Turing machine that accepts L.

I'm assuming the partial characteristic function is the same thing as an indicator function:

$1_A(x) := \{ 1 \text{ if } x \in A,$
$\quad\quad \{ 0 \text{ if } x \notin A.$

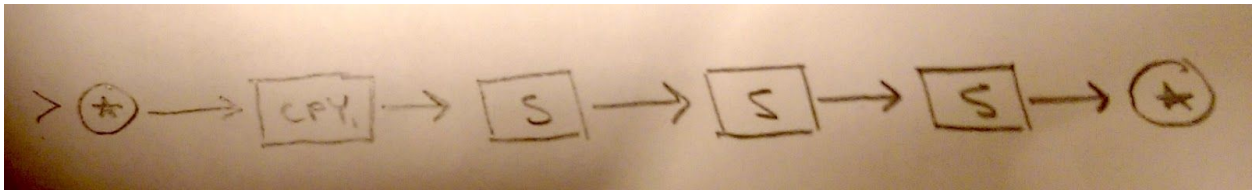
So M is supposed to recognize if a string is in the language L by halting with a 1 on the tape, and if the string is not in the language L, M will reject it by halting with a 0 on the tape. If this is the case, then I'm not sure what our machine needs to do.

4.

7. Use the macros and machines constructed in Sections 9.2 through 9.4 to design machines that computer the following functions:

(a) f(n) = 2n + 3

Assuming unary representation of n on the input tape:



10. Describe the mapping defined by each of the following composite functions:

(a) add ◦ (multi ◦ (id, id), add ◦ (id, id))
add ◦ (multi ◦ (id, id), add ◦ (id, id)) x
add ◦ (multi ◦ (id, id)(x), add ◦ (id, id)(x))
add ◦ (multi ◦ (id (x), id (x)), add ◦ (id (x), id (x))
add ◦ (multi ◦ (x, x), add ◦ (x, x))
add ◦ (a^2, 2a)
a^2 + 2a

(b) s = successor, e = erase, $p^{(n)}_m$ = pick m of n
need 2 variables: x, y
$p^{(2)}_1$ ◦ (s ◦ $p^{(2)}_1$, e ◦ $p^{(2)}_2$)
$p^{(2)}_1$ ◦ (s ◦ $p^{(2)}_1$, e ◦ $p^{(2)}_2$)(x,y)
$p^{(2)}_1$ ◦ (s ◦ $p^{(2)}_1$(x,y) , e ◦ $p^{(2)}_2$(x,y))
$p^{(2)}_1$ ◦ (s ◦ x, e ◦ y)
$p^{(2)}_1$ ◦ (x + 1, e(y))
x + 1

(c) $c^{(3)}_2$ (n_1,n_2,n_3) = 2 -> constant function
need 3 variables: x, y, z
mult ◦ ($c^{(3)}_2$, add ◦ ($p^{(3)}_1$, s ◦ $p^{(3)}_2$))
mult ◦ ($c^{(3)}_2$, add ◦ ($p^{(3)}_1$, s ◦ $p^{(3)}_2$))(x,y,z)
mult ◦ ($c^{(3)}_2$(x,y,z) , add ◦ ($p^{(3)}_1$, s ◦ $p^{(3)}_2$)(x,y,z))
mult ◦ (2, add ◦ ($p^{(3)}_1$ (x,y,z), s ◦ $p^{(3)}_2$ (x,y,z)))
mult ◦ (2, add ◦ (x, s ◦ y))
mult ◦ (2, add ◦ (x, y + 1))
mult ◦ (2, x + y + 1)
2(x + y + 1)
2x + 2y + 2

(d) need 1 variable: x
mult ◦ (mult ◦ ($p^{(1)}_1$, $p^{(1)}_1$), $p^{(1)}_1$)
mult ◦ (mult ◦ ($p^{(1)}_1$, $p^{(1)}_1$), $p^{(1)}_1$)(x)
mult ◦ (mult ◦ ($p^{(1)}_1$, $p^{(1)}_1$)(x), $p^{(1)}_1$ (x))
mult ◦ (mult ◦ ($p^{(1)}_1$ (x), $p^{(1)}_1$ (x)), $p^{(1)}_1$ (x))
mult ◦ (mult ◦ ((x, x), x))
mult ◦ (x^2, x)
x^3