Errata

Page ix, Line 13

ftp.cs.uiowa.edu

Chapter 1

Page 17, Line 7

<element> – <object>

Page 23, Line -9

… programming language, …

Chapter 2

Page 40, Line -4 and -11

touched | **surprised** …

Page 45, Line -10

Reduce font size.

Page 54, Line -15

element(E) --> [lparen], intexpr(E), [rparen].

Page 57, Line 8

<integer expr> ::= …

| **if** <boolean expr> **then** <integer expr> **else** <integer expr>

Chapter 3

Page 73, Line 16

… more divisions than multiplications,

Page 76, Under <variable>

Name: 'a'

Page 79, Line 19

then cons(head($table_1$), table-union(tail($table_1$), $table_2$))

Page 81, Line -11

lookup-type(*Name*(<variable), …

Page 82, Lines 4, 8, 13, and 14

<command sequence>.

Page 86, After Line 5

Type(<expr>) ←

lookup-type(*Name*(<variable>,*Symbol-table*(<command>))

Page 86, Line 8

integer, boolean : error("")

Page 87, Line 19 (italicize Type)

if *Type*(<element>) = *undefined*

Page 88, Line 14

if *Type*(<boolean element>) = *undefined*

Page 88, After Line 23

Type(<boolean expr>) ← *Type*(<boolean element>)

Page 89, Line 18

then cons(head($table_1$), table-union(tail($table_1$), $table_2$))

Page 89, Line -11

head([first | rest]) = first

Page 89, Line -10

tail([first | rest]) = rest

Page 89, Line -8

cons(first,rest) = [first | rest]

Page 90, Line -1

"example of a string literal"

Page 91, Header

3.2 AN ATTRIBUTE GRAMMAR FOR WREN 9 1

Page 91, Line -13

… third context conditions:

Page 102, Line 1

element([num(N)],SymbolTable,Type) --> [num(N)].

Page 102, Line 4

{ looktype(I,SymbolTable,VarType),

Chapter 4

Page 118, Line 15

DECLSEQ declaration seq, DECL declaration.

Page 129, Line -5

DECLSEQ declaration seq, DECL declaration.

Chapter 5

Page 155, Line -2 and Page 156, Line 1

(λx . (λf . f (succ x)) (λz . (λg . (λy . (add (mul (g y) x)) z))))
((λz . (add z 3)) 5)

Page 159, Line 14

a) (λf . f add (f mul (f add 5))) (λg . λx . g x x)

Page 159, Lines -5 and -6

a) Curry (Uncurry h) ⇒ h

b) Uncurry (Curry h) (Pair r s) ⇒ h (Pair r s)

Chapter 6

Page 169, Line -14

(#t (+ (fibonacci (- n 1)) (fibonacci (- n 2))))))

Page 171, Line -4

… (cadr (assoc ide env)))

Chapter 7

Page 201, Line 20

<integer expr>$_2$

Page 201, Line 21

[*TestCode*(<relation>)])

Page 204, Line 3

result

Page 204, Lines 11 and 12

temporary(*Temp*(<command>)+1)

Page 206, Line -9

[(JF.…

Page 208, Line 1

program

Page 208, Line -10

read

Page 208, Lines -4 and -5

temporary(*Temp*(<command>)+1)

Page 209, Lines 7, 17, and -13

[(JF.…

Page 210, Line 7

Temp(<boolean expr>) ← *Temp*(<expr>)

Page 211, Lines 7 and 9

temporary(*Temp*(<boolean expr>)+1)

Page 211, Lines 18 and 20

temporary(*Temp*(<boolean term>)+1)

Page 211, Line 22

Temp(<boolean element>) ← *Temp*(<boolean term>)+1

Page 211, Line -5

optimize(*Code*(<integer expr>$_2$),*Temp*(<comparison>),SUB),

Page 211, Line -4

[*TestCode*(<relation>)])

Page 214, Exercise 5

Change the semantic rules for the **write** command so that the code is optimized when the expression being printed is a variable.

Page 214, Line -6

<integer expr> ::=

if <boolean expr> **then** <integer expr>$_1$ **else** <integer expr>$_2$

Page 214, Line -3

<integer expr> ::=

begin <command sequence> **return** <integer expr> **end**

Page 221, Line -2

pretty-print predicate …

Page 222, Lines 7, 8, 9, and 11

exercises 8, 9, 10, and 11

Chapter 8

Page 234, Line 19

d) (λx . ((λy . λx . z y) x)) p (λx . x)

Page 241, Line 2

bid : bexp bid∈Bid

Page 254, Lines 3 and 4

Replace each “bie” by “be”

Chapter 9

Page 273, Line 13

Command ::= **while** Expression **do** Command^{+}

Page 276, Line 16

denotational semantics supports the substitution of ….

Page 278, Line -8

Delete **Clear** from <expression> production.

Page 280, Line -1

affects these four values.

Page 282, Lines -11, -10

updating the accumulator and the display.

Page 283, Lines 9, 10, and 11

compute ⟦+⟧ (a,op,d,m) = (op(a,d),*plus*,op(a,d),m)
compute ⟦–⟧ (a,op,d,m) = (op(a,d),*minus*,op(a,d),m)
compute ⟦**x**⟧ (a,op,d,m) = (op(a,d),*times*,op(a,d),m)

Page 284, Line -6

… *perform* ⟦E **Clear**⟧ = *perform* ⟦**Clear**⟧.

Page 285, Line 10 and 11

State = Integer x Integer x (*clear* + *unclear*), representing the display, memory, and a "clear" flag.

Page 286, Line 4

with the one in Figure 1.18 …

Page 289, Line 17

Remember that the semantic domain Store …

Page 297, Line -3

and *int*(n) = *evaluate* ⟦0⟧ $sto_{0,5}$

Page 300, Lines 10 and 11

evaluate ⟦100⟧ and *less*(22,100)

Page 303, Exercise 5

Reduce size of subscripts.

Page 303, Lines 6 and 7

e) **if** E **then** (**if** E **then** C_1 **else** C_2) **else** C_3 and **if** E **then** C_1 **else** C_3

f) (**while** E **do** C_1); **if** E **then** C_2 **else** C_3 and (**while** E **do** C_1); C_3

Page 304, Line -15

Provide a denotational definition …

Page 308, Line -10

state(Sto,Inp,Outp).

Page 309, Line -14

multiplication, **or**, …

Page 314, Line 3

var a : **integer**;

Page 316, Line -7

where env_1 = *extendEnv*(env, I_1,*proc1*(proc))

Page 319, Line 6

Section 8.2).

Page 320, Line 14

proc =
λ loc . *execute* ⟦**if** n>0 **then** s **:=** s+n**;** sum(n–1)⟧ *extendEnv*(env$_1$,n,var(loc))

Page 322, Line -14

n := 4; fac(n); **write** f

Page 322, Line -8

storable values

Page 323, Line 3

function Identifier (Identifier **:** Type) **:** Type **is** Declaration

Page 323, Exercise 10

Remove the assignment command, parameter passing, the **read** …

Also consider the values *unused* and *undefined* as the same.

Page 325, Line 1

… need to be altered …

Page 326, Line 4

check ⟦C⟧ *overlay*((*elaborate* ⟦D⟧ *emptyEnv*), env)

Page 326, Lines 7 and 8

This equation is incorrect since the expression cannot see identifiers from enclosing blocks. I have a corrected version that uses two environments in *elaborate*.

Page 337, Lines 2 and 7

perform ⟦S⟧ *emptyEnv identityCont emptySto*

and env$_1$ = *extendEnv*(env,[L$_1$, L$_2$, … , L$_n$],[cont$_1$, cont$_2$, … , cont$_n$])

Chapter 10

Page 352, Line 3

… 1≤i≤n} ∪ {⊥} with the …

Page 353, Line -4

… for i≥1 plus a new bottom element.

Page 356, Line 3

partially

Page 364, Line -4

… for all proper $n \in N$.

Page 369, Line 18

$f(1) = f(3) = f(1) = \ldots$

Page 376, Line -13

$F\ f\ d = f\ \langle F_1\ f\ d, F_2\ f\ d, \ldots, F_n\ f\ d \rangle$ for all $f \in D^n \rightarrow D$

and $d \in D^n$ is also continuous.

Page 376, Line -11

… and $F\ f\ d = f\ \langle F_1\ f\ d \rangle$ for all $f \in D \rightarrow D$ and $d \in D$.

Page 380, Line 23

$W^0 \perp \subseteq W^1 \perp \subseteq W^2 \perp \subseteq W^3 \perp \subseteq \ldots$

Page 391, Line 2

known such expression, …

Page 391, Lines 17 and 18

$\Rightarrow \quad (\lambda f\ .\ f\ ((\lambda x\ .\ f\ (x\ x))\ (\lambda x\ .\ f\ (x\ x))))\ E$

$\Rightarrow \quad (\lambda f\ .\ f\ (f\ ((\lambda x\ .\ f\ (x\ x))\ (\lambda x\ .\ f\ (x\ x)))))\ E$

Page 393, Line 19

$\Rightarrow\ 4 \bullet 3 \bullet ((\text{fix Fac})\ 2)\ \Rightarrow\ 4 \bullet 3 \bullet ((\text{Fac}\ (\textit{fix}\ \text{Fac}))\ 2)$

Chapter 11

Page 404, Line -5

{Q_1 or Q_2},

Page 410, Line -19

… and b>0 } $\supset$

Page 416, Line 2

… of two positive integers …

Page 416, Line -7

{ $M \geq 0$ and $K \geq 0$ }

Page 416, Line -5

{ result=b_K and $M = b_0 + b_1 \bullet 2 + \ldots + b_j \bullet 2^j + \ldots$ where b_j=0 or 1 }.

Page 417, Line 16

while sum<=1000 **do**

Page 417, Line -7

{ N≥2 }

Page 424, Lines 11 and -11

procedure p(f : **integer**) **is** **procedure** p(f : **integer**) **is**

Page 428, First 12 lines

Case 2: n> 0.

The recursive assumption with f=n-1 gives

{ n = K ≥ 0 and n > 0 } ⊃

{ n-1 = K-1 ≥ 0 }

factorial(n-1)

{ fact = (n-1)! and n-1 = K-1 } ⊃

{ fact = (n-1)! }

The Assign rule gives

{ fact = (n-1)! } ⊃

{ n•fact = n•(n-1)! }

fact **:=** n * fact

{ fact = n•(n-1)! = n! }, which is the desired postcondition. ∎

Page 429, Line -6

write x

Page 430, Line 6

read n; **if** n ≠ 0 **then write** n; copy **end if**

Page 430, Line 18

program multiply **is**

Page 430, Line -12

{ m = A•B }

Page 433, Line 11

P ⊃ E∈W

Page 433, Line -5

a) *(n ≥ 0 and k ≤ n and f = k! and OUT = [ ]) ⊃ (n - k ≥ 0)*

Page 434, Line 2

section 11.2

Page 436, Line 2

section 11.2

Page 436, Line 11

m := m*m;

Chapter 12

Page 447, Line 15

… definitions often suggest …

Page 467, Line 16

let B be an arbitrary model of the specification.

Page 470, Line 11

6. Consider …

Page 471, Line 1

12.3 USING ALGEBRAIC SPECIFICATIONS

Chapter 13

Page 520, Line -12

give the given Integer

Page 527, Line 10

then instead of and then

Page 529, 530, 555, and 556

Change font to serif in each value of a literal term:

value of 12

Pages 529, Lines 7 and 9

then instead of and then

Page 538, Line -12

execution of the loop body …

Page 546, Line 11

… that sets it into action.

Page 555, After line 6

var n : **integer**; -- D_4

Page 555, Line 7

procedure change **is** -- D_5

Page 555, Line 15

switch := **true**

Page 555, Line -11, -10, -9

run ⟦ **program** I **is** D_1 D_2 D_3 D_4 D_5 **begin** C_1; C_2; C_3; C_4 **end** ⟧

= elaborate ⟦ D_1 D_2 D_3 D_4 D_5 ⟧ hence execute ⟦ C_1; C_2; C_3; C_4 ⟧

= elaborate D_1 before elaborate D_2 before elaborate D_3

before elaborate D_4 before elaborate D_5

Page 555, After line -1

elaborate D_4 = allocate a cell then bind n to the given Cell

Page 556, Line 1

elaborate D_5 =

Appendix A

Page 570, Line -2

Delete X = mary;.