Typos and similar

... as they have been detected sofar

chapter 2:
→ page 18, third line from the top: wrong font → replace *is_num* by `is_num`
→ page 29, third line from the top: drop the comma immediately preceding 'which'

chapter 3:
→ page 44, underneath the first item for EVAL: there is a lost backslash \
→ page 48, third paragraph, fourth line: skip the "to" before "something"

chapter 4:
→ page 53, equation at the top: replace the = sign by ≡ (for equivalence)
→ page 63, heading of subsection 4.4: add ∗∗ to indicate it as being difficult to read
→ page 73 nine lines from the top: the definition of a head normal form should have the form

$$\lambda u_1 \ldots \lambda u_n.(\ldots(\,u\,e_1\,)\ldots e_m\,)$$

where the head variable u may either be one of the bound variables $u_1, \ldots, u_n$ or relatively free in this abstraction but bound higher up in a surrounding context.

chapter 5:
→ fig. 5.2 on page 92s, third line from the top: there is a superfluous slash /, the line must read: $([<< u\ v > < v\ w >> \ \lambda w.((u\ v)\ w)]\ u)$
→ page 95, first state transition rule six lines down: there is a superfluous ";" in front of D
→ the state transition rules in figs. 5.5 on page 97 and 5.7 on page 102 are erroneous, a corrected version can be found in this errata
→ page 107, four lines from the bottom up: `apps` must be replaced by `app`

chapter 6:
→ footnote on page 137: replace $(\ \Lambda.(e_0\ \#0)\ e_1)$ by $(\ \Lambda.(e_0'\ \#0)\ e_1)$ (added $'$ on e_0)

chapter 7:
→ page 149 third paragraph: drop the comma after "that"
→ fig. 7.3 on page 155: there is an empty symbol □ missing in the third field of the Λ node entry in the trace stack, likewise in the same node of fig. 7.5 on page 159

chapter 8:
... sofar nothing found

chapter 9:
... sofar nothing found

chapter 10:
... sofar nothing found

chapter 11:
→ page 255, last line of second paragraph: " ... reduces to the (value of the) otherwise expression e_0 ... "

chapter 12:
→ footnote on page 271: a defining equation $f \ = \ e$ turns into (`define` f e), not (f e)

→ end of first paragraph of subsection 12.1: ".. a frame in E (not C!) referenced by p_e ..."

chapter 13:
→ page 313, there is a superfluous = sign in the second line of the syntax definition for procedures, it must read: *procedure* $=_s$ `sub` $form\_pars$ `in` *body*
→ there is an unfortunate page break at the bottom of page 315: the last two lines, beginning at "... where $h \in \{1, \ldots, k\}, \ldots$ belong between the two code fragments on the next page, before the line $\underbrace{\Delta_g \ldots \Delta_g}_{k}$ `in`

chapter 14:
→ on page 330, in the middle of the first paragraph of subsection 14.1.4: an "x" has slipped into the word "provisions"

→ page 337, last line of third paragraph: replace "procedure" by procedures"

appendix A:
→ page 350, the first rule of the input/output scheme at the bottom: the righthand side must read: $C[\, main[\, u \leftarrow resp\,]\,] \;\|\; < in\ out >$

references:
→ reference [Dyb87]: the full name of the author is Dybvig, not Dybvik.

Corrected specification of the SE(M)CD machine

There are two errors in the specification of the SE(M)CD machine in chapter 5, figs.5.5 and 5.7. One relates to return continuations saved on (and unsaved from) the dump, which should also include stack M. The other is more severe as it creates wrong environments for closures / suspensions of arguments under normal order evaluation.

The following table gives a corrected and otherwise improved specification of the #SE(M)CD machine for the nameless Λ-calculus, based on the syntax

$$e \;=_s\; \#j \mid \overline{@}\, e_a\, e_f \mid @\, e_f\, e_a \mid \Lambda\, e_b \mid [\, E\; e\,]\;,$$

i.e., legitimate expressions that can be processed by the machine are deBruijn indices, applicative and normal order applications, nameless abstractions, and closures / suspensions. Whenever a rule applies to both the $\overline{@}$ and @ applicators, they are summarily denoted as *ap*.

The state transition function is again defined as

$$\tau : (S,\; E,\; M,\; C,\; D) \;\rightarrow\; (S',\; E',\; M',\; C,\; D)\;\;.$$

Rules $1a/b$ use deBruijn indices popping to the top of the code structure C to access the environment and push the associated entries on top of S. Rules $3a/b$ split applications on top of C up into their components, pushing the apply nodes on top of M and (re)arranging the subexpressions on C so that the operand in both cases precedes the operator. Rule 2 identifies, by means of the arity index associated with an apply node @ on top of M, the top expression on C as operand of a normal order application, creates a closure for it and pushes it into S. Rules $4a/b$ create closures on S for abstractions on top of C.

Rules $5a/b$ intercept instances of naive β-reductions as configurations with an abstraction closure on top of an operand in S and an apply node with arity 0 on top of M. Both rules isolate the abstraction body in C, prepend the operand as new entry to the environment E, and save a return continuation in D. Rule 6 effects evaluation of a closure / suspension created by rule 3 for an application in operand position.

Rules $7a/b$ and $8a/b$ reconstruct in S irreducible applications.

Rule 9 retrieves from the dump a return continuation upon completion of a β-reduction.

$$(1a)\ \ (S,\ E,\ nil,\ \#j : C,\ D)$$
$$\rightarrow\ (lookup(\ \#j,\ E\ ) : S,\ E,\ nil,\ C,\ D)$$
$$(1b)\ \ (S,\ E,\ ap^{(i)} : M,\ \#j : C,\ D)\ |\ (i > 0)$$
$$\rightarrow (lookup(\ \#j,\ E\ ) : S,\ E,\ ap^{(i-1)} : M,\ C,\ D)$$

$$(2)\ \ (S,\ E,\ @^{(2)} : M,\ e_a : C,\ D)$$
$$\rightarrow ([\ E\ e_a\ ] : S,\ E,\ @^{(1)} : M,\ C,\ D)$$

$$(3a)\ \ (S,\ E,\ M,\ \overline{@}\, e_a\, e_f : C,\ D))$$
$$\rightarrow (S,\ E,\ \overline{@}^{(2)} : M,\ e_a : e_f : C,\ D)$$
$$(3b)\ \ (S,\ E,\ M,\ @\, e_f\, e_a : C,\ D)$$
$$\rightarrow (S,\ E,\ @^{(2)} : M,\ e_a : e_f : C,\ D)$$

$$(4a)\ \ (S,\ E,\ nil,\ \Lambda\, e_b : C,\ D)$$
$$\rightarrow ([\ E\ \Lambda\, e_b\ ] : S,\ E,\ nil,\ C,\ D)$$

$$(4b)\ \ (S,\ E,\ ap^{(i)} : M,\ \Lambda\, e_b : C,\ D)\ |\ (i > 0)$$
$$\rightarrow ([\ E\ \Lambda\, e_b\ ] : S,\ E,\ ap^{(i-1)} : M,\ C,\ D)$$

$$(5a)\ \ ([\ E'\ \Lambda\, e_b\ ] : e_a : S,\ E,\ ap^{(0)} : nil,\ C,\ D)$$
$$\rightarrow\ (S,\ e_a : E',\ nil,\ e_b : nil,\ (E,\ nil,\ C,\ D))$$
$$(5b)\ \ ([\ E'\ \Lambda\, e_b\ ] : e_a : S,\ E,\ ap^{(0)} : ap^{(i)} : M,\ C,\ D)$$
$$\rightarrow\ (S,\ e_a : E',\ nil,\ e_b : nil,\ (E,\ ap^{(i-1)} : M,\ C,\ D))$$

$$(6)\ \ ([\ E'\ e_a\ ] : S,\ E,\ M,\ C,\ D)\ |\ e_a\ =\ ap\, e_0\, e_1$$
$$\rightarrow\ (S,\ E',\ nil,\ e_a : nil,\ (E,\ M,\ C,\ D))$$

$$(7a)\ \ (e_0 : e_1 : S,\ E,\ \overline{@}^{(0)} : nil,\ C,\ D)$$
$$\rightarrow\ (\overline{@}\, e_1\, e_0 : S,\ E,\ nil,\ C,\ D)$$
$$(7b)\ \ (e_0 : e_1 : S,\ E,\ \overline{@}^{(0)} : ap^{(i)} : M,\ C,\ D)\ |\ (i > 0)$$
$$\rightarrow\ (\overline{@}\, e_1\, e_0 : S,\ E,\ ap^{(i-1)} : M,\ C,\ D)$$

$$(8a)\ \ (e_0 : e_1 : S,\ E,\ @^{(0)} : nil,\ C,\ D)$$
$$\rightarrow\ (@\, e_0\, e_1 : S,\ E,\ nil,\ C,\ D)$$
$$(8b)\ \ (e_0 : e_1 : S,\ E,\ @^{(0)} : ap^{(i)} : M,\ C,\ D)\ |\ (i > 0)$$
$$\rightarrow\ (@\, e_0\, e_1 : S,\ E,\ ap^{(i-1)} : M,\ C,\ D)$$

$$(9)\ \ (S,\ E,\ nil,\ nil,\ (E',\ M',\ C',\ D')) \rightarrow (S,\ E',\ M',\ C',\ D')$$