2005-9-7 p. 235 precedence of √ moved from level 12 to level 2

2005-9-15 p.235 change “means” to “means the same as” 3 times.


2005-9-26 p.0 “A closely related theory is Dijkstra's” to “A closely related theory uses Dijkstra's”.

2005-9-26 p.34 change “Our example state space is infinite” to “Our example state space in the previous paragraph is infinite”.

2005-9-26 p.40 change “In each, the problem (left side) follows from the solution (right side)” to “In each, the problem (left side) is refined by (follows from, is implied by) the solution (right side)”.

2005-10-18 p.45 Exercise 45 is now called “Cantor's paradise”. Also p.215.

2005-10-4 p.-1 increase the section number of 10.0 to 10.8 by 1, then add 10.0 Preface and make the same changes in Chapter 10.


2005-11-19 p.54 and p.55 the hint “expand assignment” occurs 3 times, and each is wrong. The first occurrence is on p.54, and it should be changed to the following.

\[(h<j) \Rightarrow R \iff j-h = 1 \land (\text{p}:=Lh=x)\]

make the colon plain (not italic).

2005-11-2 p.53 change the two sentences “Surprisingly, ... rather than two.” to “According to the recursive measure, the worst case time is not improved at all, and the average time is improved slightly by a factor of \((#L)/(#L+1)\) assuming equal probability of finding the item at each index and not finding it at all. And according to the real time measure, both the worst case and average execution times are a lot worse because the loop contains three tests rather than two.”.

2005-11-19 p.54 and p.55 the hint “expand assignment” occurs 3 times, and each is wrong. The first occurrence is on p.54, and it should be changed to the following.

\[(j-hz2) \Rightarrow h'=h<\text{i'}<j'=j' \iff i:=\text{div}(h+j)2\]

The second occurrence is on p.55 and it should be changed to the following.

\[(j-hz2) \Rightarrow h'=h<\text{i'}<j'=j' \iff i:=\text{div}(h+j)2 \land p'=p \land h'=h \land j'=j\]

The last occurrence is on p.55 and it should be changed to the following.
(U ⇐ j−h = 1 ∧ (p:=Lh=x))
expand U and the assignment
= (h<j ⇒ t′ ≤ t + ceil (log (j−h))) ⇐ j−h=1 ∧ p′=(Lh=x) ∧ h′=h ∧ i′=i ∧ j'=j ∧ t'=t)
use main antecedent as context in main consequent
= (h<j ⇒ t ≤ t + ceil (log 1) ⇐ j−h=1 ∧ p′=(Lh=x) ∧ h′=h ∧ i′=i ∧ j'=j ∧ t'=t)
Use log 1 = 0
= (h<j ⇒ T ⇐ j−h=1 ∧ p′=(Lh=x) ∧ h′=h ∧ i′=i ∧ j'=j ∧ t'=t)
base law twice
= T

--------------------------

p.53 “introduce variable j” becomes “introduce natural variables i and j”

--------------------------

p.55 replace first existentially quantified line and following line by

(∃h'', i'', j'', p'': h''=h<i''<j'' ∧ Li''≤x
∧ (i''<j'' ⇒ (x: L ((i''..j'') = p' ⇒ Lh′ = x)))
⇒ R)
eliminate p'', h'', and j'' by one-point, and rename i'' to i

--------------------------

p.104 near bottom: change “two of those items may be lists themselves” to “two of those items
are lists themselves”

--------------------------

p.108 middle becomes:
The axioms use an auxiliary specification that helps in writing the axioms, but is not an addition to
the theory, and does not need to be implemented: work means “Do anything, wander around
changing the values of nodes if you like, but do not go from this node (your location at the start of
work ) in this direction (the value of variable aim at the start of work ). End where you started,
the first design decision might be to divide the task into several pieces that will fit together in some
way. This decision can be written as a refinement, specifying exactly what the parts are and how
they fit together, and then the refinement can be proven. Using the theory in the early stages is
enormously beneficial, because if an early step is wrong, it is enormously costly to correct later.

For a theory of programming to be in widespread use for industrial program design, it must be
supported by tools. Ideally, an automated prover checks each refinement, remaining silent if the
refinement is correct, complaining whenever there is a mistake, and saying exactly what is wrong.
At present there are a few tools that provide some assistance, but they are far from ideal. There is
plenty of opportunity for tool builders, and they need a thorough knowledge of a practical theory of
programming.

--------------------------

p.16 on the line after
insert “where \( i \) is an integer.”. And on p. 227 change the two lines

\[
i: x,..,y = x \leq i < y
\]

(\( \phi(x,..,y) = y-x \))

(\( \text{for extended integers } i, x, y, x \leq y \))

\( x, y: \text{xint} \land x \leq y \Rightarrow \phi(x,..,y) = y-x \)

---

p. 52 mid change paragraph as follows:

Refinement by Parts says that if the same refinement structure can be used for two specifications, then it can be used for their conjunction. If we add \( t := t+1 \) to the refinements that were not concerned with time, it won't affect their proof, and then we have