Errata for
Synchronization Algorithms and Concurrent Programming

Gadi Taubenfeld*

November 19, 2017

In the first edition of a book with more than 400 pages, a couple of typos are bound to slip in. The errata below list the mistakes that I will fix in the next printing of the book. If you see errors not noted below, please send me mail at: tgadi@idc.ac.il. Please feel free to contact me with any criticism or comments which might help to improve any future version of this book. I would be glad to hear from you!

Visit the Companion Website at http://www.faculty.idc.ac.il/gadi/book.htm to find valuable online resources.

*The Interdisciplinary Center, P.O.Box 167, Herzliya 46150, Israel. tgadi@idc.ac.il
Chapter 1

p.4  lines 5-6, “can not” should be “cannot”
p.7  lines 3, “than” should be “then”
p.8  line 3, “can not” should be “cannot”; line 14: “assumption” should be “assumptions”
p.12 line 4, “sometime” should be “sometimes”; lines -8,-13: “can not” should be “cannot”
p.14 Informally, absence of contention means that no other process is in the midst of
preforming it entry/cs/exit code. line -1: “time” should be “times”
p.17 line -8, “issue” should be “issues”
p.21 line -10, “all variables are” should be “variable x is”
p.23 line 13, “briefly only” should be “only briefly”

Chapter 2

p.33 Figure 2.1, in the oval, $b[i]$ should be $b[j]$
p.37 line 6, construction of
p.38 line # 11 of the algorithm, should be “$node := \lfloor i/2^{level} \rfloor$”
p.43 line 3-4: remove “, or the critical section is occupied”
p.44 line 4, “in absence” should be “in the absence”; line 24: remove “and the”
p.56 line # of the algorithm, false in italic
p.58 line -9, remove “(1) No”, and renumber the 4 remaining answers
p.67 line -6, replace “never” with “always”
p.69 line 10, “x or” should be “xor”
p.75 line 10, “1.20(” should be “1.20,)”
p.79 line 6, replace “k” with “turn”
p.85 problem 2.42, line #4 of the algorithm should be “await $number[j] \neq -1$”
p.92 line 3 (line # 9 of the algorithm), “if” should be “fi”

Chapter 3

p.107 Figure 3.3, it says “await(b = 0 or z = 1)” and should be “await(b = 1 or z = 1)”
p.112 Figure 3.5, the direction of the bottom arrows should reversed
p.120 line 8, “then” should be “than”
p.139 line -6, delete the “a”
p.142 line #1 of the algorithm, “%” should be “/”

Chapter 4

p.155 line #4 of the algorithm “=” should be “\neq”
p.173 line D4, “lhead” should be “lhead.ptr”
  lines E13, E17, D13: “O” should be “Q”
p.177 line 9, “general operation” should be “general semaphore”
p.189 line -17, delete the word “the”
p.192 line 11, delete the word “for”
p.200 line $ of the algorithm, move comment to the right side
Chapter 5

The power-point presentation at the companion website (http://www.faculty.idc.ac.il/gadi/book.htm) includes the full correct versions of the algorithms.

p. 205 In line #6 of version #2 of the algorithm, “\texttt{await(local.go \neq go)}” should be \texttt{\texttt{await(local.go = go)}}

p.205-206 In the three algorithms in Sections 5.2.1 and 5.2.2, replace lines 2 and 3 with:
2 \texttt{local.counter := fetch-and-increment(counter)}
3 \texttt{if local.counter + 1 = n then}

p.207 Replace line 6 of the algorithm with:
6 \texttt{then go := 1 - local.go fi}

p.208-209 In both algorithms in Section 5.3, \texttt{counter} should be defined as a test-and-test-and-set bit, and line 5 should be replaced with:
6 \texttt{\texttt{await (counter = 1) fi}}

p.209 lines 10,11,12 of the algorithm should be renumbered as 9,10,11, respectively.

p.211 In the algorithm in Section 5.4, replace lines 2 and 3 with:
2 \texttt{local.counter := fetch-and-increment(counter[level,node])}
3 \texttt{if local.counter + 1 = degree then}

p.212-213 In the algorithm, replace \texttt{go[1..n]} with \texttt{go[2..n]}
In lines: 4, 10, and 13 of the algorithms, in the assignments, use = instead of :=

p.213 line –9, “from process $i + 2^r \pmod{n}$” should be “from process $i - 2^r \pmod{n}$”

p.213 last line, “from process $i + 2^r \pmod{n}$” should be “from process $i - 2^r \pmod{n}$”
In the algorithm, the initial value of \texttt{sense} is 1 (instead of 0)

p.217 Update Rule 5 with as follows:
\textbf{Rule 5:} Applicable if scheduled process notices that the go bit has been flipped (relative to its local.go, that is, \texttt{go \neq local.go}).
The process knows that everybody has arrived and continues past the barrier

p.217-218 See next page for an updated version of the See-Saw Barrier.

p.218 \textit{Token invariant:} Assume that at the beginning of an episode of the see-saw barrier the state of each process is \texttt{never-been-on}. Then, until the go bit is flipped, the number of tokens in the system is either $2n$ or $2n + 1$.
\textit{Balanced invariant:} Assume that at the beginning of an episode of the see-saw barrier the state of each process is \texttt{never-been-on}. Then, until the go bit is flipped, the number of the left and right side of the see-saw is either perfectly balanced or favors the down-side of the see saw by one process.

p.219 line 2, $2^k$ should be $k$.
The semaphores \texttt{arrive1} and \texttt{arrive2} are initially both 0 (and not 1)

p.220 In lines 2 & 5 of the first alg. and lines 2 & 5 of the second alg., = should be :=

p.221 line 5: “There dozens” should be “There are dozens”

p.224 “5.15 ... algorithm” should be “5.15 ... algorithms”

p.225 line 10 of the algorithm, \texttt{counter = n} should be \texttt{counter = 0}
The code of the See-Saw Barrier. We use *token*, *see-saw* and *go* to designate the first, second and third components, respectively, of the ordered triple stored in the 8-valued RMW register (3 bits). We emphasize that accessing the RMW register is done in one atomic action.

THE SEE-SAW BARRIER: program of a process  /* there are n processes */

```plaintext
<table>
<thead>
<tr>
<th>line</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>local.go := go  /* remember current value (a RMW operation) */</td>
</tr>
<tr>
<td>2</td>
<td>mystate := never-been-on</td>
</tr>
<tr>
<td>3</td>
<td>mytokens := 2  /* enters with two tokens */</td>
</tr>
<tr>
<td>4</td>
<td>repeat</td>
</tr>
<tr>
<td>5</td>
<td>if local.go ≠ go then mystate := got-off  /* last with one token? */</td>
</tr>
<tr>
<td>6</td>
<td>elseif mystate = never-been-on then</td>
</tr>
<tr>
<td>7</td>
<td>if see-saw = left-side-down then  /* gets on the up-side */</td>
</tr>
<tr>
<td>8</td>
<td>mystate := on-right-side</td>
</tr>
<tr>
<td>9</td>
<td>else mystate := on-left-side</td>
</tr>
<tr>
<td>10</td>
<td>if see-saw = left-side-down then  /* gets on the up-side */</td>
</tr>
<tr>
<td>11</td>
<td>elseif token = no-token-present and  /* token bit empty? */</td>
</tr>
<tr>
<td>12</td>
<td>if mystate = on-left-side and see-saw = right-side-down then</td>
</tr>
<tr>
<td>13</td>
<td>elseif token = no-token-present and  /* token bit empty? */</td>
</tr>
<tr>
<td>14</td>
<td>see-saw := R(see-saw) fi</td>
</tr>
<tr>
<td>15</td>
<td>mystate := got-off  /* gets off the See-Saw */</td>
</tr>
<tr>
<td>16</td>
<td>see-saw := R(see-saw) fi</td>
</tr>
<tr>
<td>17</td>
<td>mystate := got-off  /* gets off the See-Saw */</td>
</tr>
<tr>
<td>18</td>
<td>see-saw := R(see-saw) fi</td>
</tr>
<tr>
<td>19</td>
<td>elseif mystate = got-off  /* gets off the See-Saw */</td>
</tr>
<tr>
<td>20</td>
<td>elseif token = token-present and  /* token bit full? */</td>
</tr>
<tr>
<td>21</td>
<td>if mystate = on-right-side and see-saw = left-side-down then</td>
</tr>
<tr>
<td>22</td>
<td>elseif token = token-present and  /* token bit full? */</td>
</tr>
<tr>
<td>23</td>
<td>see-saw := R(see-saw) fi</td>
</tr>
<tr>
<td>24</td>
<td>elseif (mytokens ≥ 2n) or</td>
</tr>
<tr>
<td>25</td>
<td>(mytokens = 2n − 1 and token = token-present) then  /* have arrived? */</td>
</tr>
<tr>
<td>26</td>
<td>go := 1 − local.go  /* notifies all */</td>
</tr>
<tr>
<td>27</td>
<td>mystate := got-off  /* gets off the See-Saw */</td>
</tr>
<tr>
<td>28</td>
<td>until (mystate = got-off)</td>
</tr>
<tr>
<td>29</td>
<td>await (local.go ≠ go)  /* a RMW operation; Rule 5: End */</td>
</tr>
</tbody>
</table>
```

R(·) is the reflection function on {left-side-down, right-side-down}.

The See-Saw Barrier program of a process/* there are n processes */

```plaintext
type token.states = ranges over {token-present, no-token-present}
see-saw.states = ranges over {left-side-down, right-side-down}
shared (token, see-saw, go): RMW ranges over token.states × see-saw.states × {0,1}
local mystate: 4-valued register, ranges over {never-been-on, on-left-side, on-right-side, got-off}
local.go: bit, ranges over {0,1}
```

R(·) is the reflection function on {left-side-down, right-side-down}.

The See-Saw Barrier program of a process/* there are n processes */

```plaintext
type token.states = ranges over {token-present, no-token-present}
see-saw.states = ranges over {left-side-down, right-side-down}
shared (token, see-saw, go): RMW ranges over token.states × see-saw.states × {0,1}
local mystate: 4-valued register, ranges over {never-been-on, on-left-side, on-right-side, got-off}
local.go: bit, ranges over {0,1}
```

R(·) is the reflection function on {left-side-down, right-side-down}.
Chapter 7

p.263  line -9, “a L-type” should be “an L-type”

p.264  line 7, “its” should be “his”

p.270  line -5, “By 7.16” should be “By Lemma 7.16”

p.272  line 4, “it” should be “he” (twice)

p.273  line 6, “left” should be “left” (i.e., italic)

Chapter 8

p.288  line 19, remove “the”

p.290  line -7, “the new” should be “some”

Chapter 9

p.309  line -3, “p_y” should be “p_i”

p.324  Figure 9.4, “proofs” should be “proof”; (a), (b), and (c) should be in roman type

p.338  line 14 (declaring turn), “values” should be “value”

add then, at the end of line #4 of the program for process 1

Bibliography

p.373  ref. [1], “per registers” should be “per register”

p.376  ref. [28], “J. Anderson” should be “J.H. Anderson”

p.389  ref. [127], “Gary” should be “Gray”

p.394  ref. [169], “page 522” should be “page 522-529”

p.397  ref. [192], “LNCS 674” should be “LNCS 647”

Add the following new paragraph at the end of Page xiv:

Acknowledgements

Thanks to all who have sent us errata to improve this book, including: Yehuda Afek, Itai Avrian, Angelo Borsotti, Peter A. Buhr, Kai Engelhardt, Denis Golyanov, Danny Handler, Frédéric Haziza Shachar Gidron, Marios Mavronicolas, Yoram Moses, Francisco Solsona, Edward Strassberger, Michel Raynal.