Errata for

Subrecursive Programming Systems: Complexity and Succinctness, by James S. Royer and John Case, Birkhäuser Boston, 1994.

James S. Royer[*] John Case[†]

March 29, 2013

The Key Problem

Lemma 3.8(b) of [RC94] asserts an inequality

$$\Phi^{\mathrm{TM}}_{norm(p)} \leq \lambda x.a \cdot |p| \cdot (\Phi^{\mathrm{TM}}_p(x) + 1),$$

but later arguments in Chapter 3 assume that the right-hand side of this inequality is $\lambda x.a \cdot (\Phi^{\mathrm{TM}}_p(x) + 1)$, and worse, those later arguments do not seem to work when one uses the actual inequality of Lemma 3.8(b).

In Figure 1 just below we indicate some significant dependencies of later results on Lemma 3.8(b).

$$\begin{tikzcd}
 & \text{Lemma 3.8(b)} \arrow[dl] \arrow[dr] & \\
\text{Lemma 3.9} \arrow[d] & & \text{Lemmas 3.10, 3.13–3.18} \arrow[d] \arrow[ddl] \\
\text{Theorem 4.16} & & \text{Theorem 4.7} \arrow[dl] \\
 & \text{Theorem 4.8} &
\end{tikzcd}$$

Figure 1: Key Dependences on Lemma 3.8(b)

In the next section we present our repair of the above indicated key problem with [RC94].

[*]Dept. of Elec. Eng. and Computer Science, Syracuse University, Syracuse, NY, 13244, USA

[†]Dept. of Computer and Information Sciences, University of Delaware, Newark, DE 19716 USA

A Repair

To correct this problem we make a minor, and fairly natural, revision in our notion of Turing Machine program. Specifically, we now allow "don't-care" markers in both the symbols-read and symbols-written k-tuples as detailed below. The effect of this change is that certain Turing Machine programming tasks can be done much more concisely. This new conciseness supports a simple, low-complexity construction for a revised Lemma 3.8 that includes the $\Phi^{\mathrm{TM}}_{norm(p)} \leq \lambda x.a \cdot (\Phi^{\mathrm{TM}}_p(x) + 1)$ inequality. With this change, the later proofs in Chapter 3 and onward now work as intended.

Below we describe the changes that need to be made to the text and provide the new version of Lemma 3.8 along with its proof.

The Changes

Page 32, second and third bullets. These are changed to:

- the *k-tuple of symbols read* by the TM's k heads—a k-tuple of elements from $\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}, *\,\}$,
- the *k-tuple of symbols written* by the TM's k heads—a k-tuple of elements from $\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}, *\,\}$,

The $*$ is the "don't-care" marker. In the symbols-read k-tuple, $\mathbf{a}_i = *$ means any character matches the $*$. In the symbols-written k-tuple, $\mathbf{b}_i = *$ means the character in the square scan by the i-th tape-head is not changed.

Page 32, last line. Change "the i-th tape head is reading $\mathbf{a}_i$" to "the i-th tape head is reading $\mathbf{a}_i$ or $\mathbf{a}_i = *$".

Page 33, line 1. Change "must necessarily have $\mathbf{a}_i = \mathbf{B}$" to "must necessarily have $\mathbf{a}_i = \mathbf{B}$ or $\mathbf{a}_i = *$".

Page 33, lines 10–14. Change these lines to:

Suppose $\mathbf{c}_2$ is the symbol read by the output-tape head. Then:
  if $\mathbf{b}_2 = *$, then write $\mathbf{c}_2$ on the output tape;
  if $\mathbf{b}_2 = \mathbf{B}$ and $\mathbf{d}_2 \neq \mathbf{N}$, then write $\mathbf{0}$ on the output tape *(per Convention 4 above);*
  otherwise, write $\mathbf{b}_2$ on the output tape.
For $i = 3, \ldots, k$, suppose $\mathbf{c}_i$ is the symbol read by tape i's head. Then:
  if $\mathbf{b}_i = *$, then write $\mathbf{c}_i$ on tape i, otherwise, write $\mathbf{b}_i$ on the tape i.

Page 33, lines -9 and -5. Change "$\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}, \mathbf{L}, \mathbf{R}, \mathbf{N}\,\}$" to "$\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}, \mathbf{L}, \mathbf{R}, \mathbf{N}, *\,\}$" both places.

Page 33, displayed equation (1). Change the contents of this to:

$$\begin{array}{lll} \tau(\mathbf{B}) = \tau(\mathbf{L}) = \mathbf{010} & \tau(\mathbf{0}) = \tau(\mathbf{R}) = \mathbf{000} & \tau(\mathbf{1}) = \tau(\mathbf{N}) = \mathbf{001} \\ \tau(*) = \mathbf{011} & \tau(\mathbf{a}_1 \ldots \mathbf{a}_n) = \tau(\mathbf{a}_1) \ldots \tau(\mathbf{a}_n) & \end{array}$$

Page 33, displayed equation (2). Change the right-hand side of this to:

$$\mathbf{111}\tau(\mathit{state})\mathbf{111}\tau(\mathbf{a}_1 \ldots \mathbf{a}_k)\mathbf{111}\tau(\mathbf{b}_1 \ldots \mathbf{b}_k)\mathbf{111}\tau(\mathbf{d}_1 \ldots \mathbf{d}_k)\mathbf{111}\tau(\mathit{next})\mathbf{111}$$

Page 34, line 1. Change $\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}\,\}$ to $\{\,\mathbf{B}, \mathbf{0}, \mathbf{1}, *\,\}$.

Page 34, lines 11–12. Change

> a fixed TM program that, on any input, does nothing but halt in 0 steps.

to

> a fixed two-tape TM program with a program consisting of the single instruction $(0, (*, *), (*, *), (\mathbf{N}, \mathbf{N}), 1)$; hence, on all inputs, this machine halts in one step with output 0.

Page 39, lines 1–17 (the proof of part (m)). Straightforwardly change this to conform to the new coding of TM programs.

Pages 40–45 (the proof of Theorem 3.6). Straightforwardly change this to conform to the changes in TM programs and their coding.

Page 45, the statement of Lemma 3.8, line 1. Change "$a \in N$" to "$a > 0$".

Page 45, the statement of Lemma 3.8. Change part (b) to:

> (b) $\Phi^{\mathrm{TM}}_{norm(p)} \leq \lambda x.a \cdot (\Phi^{\mathrm{TM}}_{p}(x) + 1)$.

Page 47, the proof of Lemma 3.8. Here is the revised argument.

Proof. Let q_0 be the TM-program that, by our conventions, all abnormal codes name. (Recall that TM_{q_0} has two tapes and has a program consisting on the single instruction $(0, (*, *), (*, *), (\mathbf{N}, \mathbf{N}), 1)$. Thus, on all inputs, TM_{q_0} halts in one step in state 1.)

For p, an abnormal code, define $norm(p) = q_0$. For p, a normal code, define $norm(p)$ to be the code of a TM M_p given below.

Let k be the number of tapes of TM_p. M_p then has $k + 1$ tapes. The first k tapes are used to run TM_p and the $(k + 1)$-st tape keeps track of run time of TM_p, recorded as a string of $\mathbf{0}$'s. M_p's program consists of three parts. PART 1: This part is a version of TM_p's program modified so that: (a) each state number in TM_p's instructions is multiplied by 2 and (b) each instruction, in addition to its TM_p actions, also writes a $\mathbf{0}$ on tape $k + 1$ and moves right on that tape. (Note, since $2 \cdot 0 = 0$, the start state 0 is not changed by the above.) PART 2: This part consists of a block of instructions of the form

$$(2 \cdot s, (*, \ldots, *, *), (*, \ldots, *, *), (\mathbf{N}, \ldots, \mathbf{N}, \mathbf{L}), 3)$$

for each state s of TM_p. We make sure that this block occurs following the instructions of part 1 in M_p's sequence of instructions. The instructions in this block catch what would otherwise be halting steps in executions of part 1 of M_p's program. (Recall that if there are multiple instructions that could apply at a point in a TM computation, the leftmost such instruction in the program is followed.) Each of these instructions also positions the head of tape $k+1$ of the rightmost **0** on that tape (if any), and transfers control to the beginning of part 3. PART 3: This part is a clean-up subroutine (with beginning-state 3) that uses the string of **0**'s on tape $k+1$ as a "ruler" to erase all the nonblank symbols on tapes 3 through $k+1$, after which it goes into state 1 (to halt). (The subroutine assumes that if tape $k+1$ is not blank, then that tape's head is positioned on the rightmost **0**.) The eight instructions of Figure 2 below suffice for the subroutine. (Note that none of these instructions change the contents of or positions on tapes 1 and 2.)

$$
\begin{array}{l}
(3,(*,\ldots,*,\mathbf{0}),(*,\ldots,*,\mathbf{0}),(\mathbf{N},\ldots,\mathbf{N}),5)\\
(3,(*,\ldots,*,\mathbf{B}),(*,\ldots,*,\mathbf{B}),(\mathbf{N},\ldots,\mathbf{N}),1)\\
(5,(*,\ldots,*,\mathbf{0}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{0}),(\mathbf{N},\mathbf{N},\mathbf{L},\ldots,\mathbf{L},\mathbf{L}),5)\\
(5,(*,\ldots,*,\mathbf{B}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{B}),(\mathbf{N},\mathbf{N},\mathbf{R},\ldots,\mathbf{R},\mathbf{R}),7)\\
(7,(*,\ldots,*,\mathbf{0}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{0}),(\mathbf{N},\mathbf{N},\mathbf{R},\ldots,\mathbf{R},\mathbf{R}),7)\\
(7,(*,\ldots,*,\mathbf{B}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{B}),(\mathbf{N},\mathbf{N},\mathbf{R},\ldots,\mathbf{R},\mathbf{L}),9)\\
(9,(*,\ldots,*,\mathbf{0}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{B}),(\mathbf{N},\mathbf{N},\mathbf{R},\ldots,\mathbf{R},\mathbf{L}),9)\\
(9,(*,\ldots,*,\mathbf{B}),(*,*,\mathbf{B},\ldots,\mathbf{B},\mathbf{B}),(\mathbf{N},\mathbf{N},\mathbf{N},\ldots,\mathbf{N},\mathbf{N}),1)
\end{array}
$$

Figure 2: Instructions for Part 3 of M_p's program

(If $\mathbf{0}^t$ is the contents of tape $k+1$ at the start of the subroutine's run, then the above instructions take each of the heads on tapes 3 through k and moves them t-many squares to the left, erasing as they go, and then moves these heads $2t$-many squares to the right, erasing as they go. They also erase tape $k+1$ on its final use as a ruler.) *Note:* M_p contains no instruction of the form

$$(1,(\mathbf{a}_1,\ldots,\mathbf{a}_k),(\mathbf{b}_1,\ldots,\mathbf{b}_k),(\mathbf{d}_1,\ldots,\mathbf{d}_k),s).$$

So if M_p ever enters state 1, it halts.

Claim 1: On each input x, the machine M_p computes the same output as TM_p and halts if and only if it does so in state 1 with all of its work tapes blank.

Proof: This follows from our description of TM_p's program.

Claim 2: The run time of M_p on input x is $\leq 4 \cdot \Phi_p^{\mathrm{TM}}(x) + 5$.

Proof: This follows by a straightforward analysis of the run time of our sketch of M_p, which analysis we leave to the reader.

Claim 3: One can compute the code of M_p in $\mathcal{O}(|p|)$ time.

Proof: It is a straightforward construction to build parts 1 and 2 of M_p's program from TM_p's in time linear in $|p|$. The task of amalgamating parts 1 and 2 with the instructions given above for part 3 can straightforwardly done in in $\mathcal{O}(|p|)$-time.

By Lemma 3.2(m), deciding the normality of a TM code can be done in $\mathcal{L}time$. Hence, by Claim 3 it follows that $norm \in \mathcal{L}time$.

Parts (a) and (c) of the lemma follow Claim 1 and our choice of q_0.

Part (b) follows by Claim 2 and our choice of q_0. ☒

References

[RC94] James S. Royer and John Case, *Subrecursive programming systems: Complexity and succinctness*, Research monograph in *Progress in Theoretical Computer Science*, Birkhäuser Boston, 1994.