Corrections as of February 14, 2000 in: **Computability and Complexity from a Programming Perspective**

Neil D. Jones

DIKU, Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
E-mail: `neil@diku.dk`

Global Remarks

- **isomorphism** ⇒ **bijection** many places!
- Page xiv item 3: **bootsrapping** ⇒ **bootstrapping**

Chapter 1

- Section 1.5.2 line 1: **to be able** ⇒
- Page 25 line -2: **statemnt** ⇒ **statement**

Chapter2

1. Figure 2.1: `b b` ⇒ `b e`
2. Page 30, line 3 and line before 2.1.3: remove 3 × `;`
3. Page 32, line -2: **Aabbreviation** ⇒ **Abbreviation**
4. Page 33, line 3: `atom?cons?` ⇒
5. Page 34: remove 3 × `;` after `write Y`
6. Page 35 l. 11: $(\mathtt{d}_1\ \mathtt{d}_2 \ldots \mathtt{d}_{n-1}\ \mathtt{d}_n\ \mathtt{e}_1\ \mathtt{e}_2 \ldots \mathtt{e}_m)$ ⇒ $(\mathtt{d}_1\ \mathtt{d}_2 \ldots \mathtt{d}_{n-1}\ \mathtt{e}_1\ \mathtt{e}_2 \ldots \mathtt{e}_m)$
7. Page 35 l. -10: remove `;` after `write Y`
8. Page 35 l. -5: copy `pp` of `p` ⇒ copy `pp` of the body of `p`
9. Page 40: remove 2 × `;` after `write Y`

10. Page 41 l. 4: remove 4 × ; after `C_`'s

11. Page 34: remove 3 × ; after `write Y`

12. Page 42 l. -7: add ; after `X := E`

13. Page 43, l. -11: BUG, since `case` uses `=?` ! (Easy to fix, since `case` only compares with constants (`=? E d`): Expand into a series of `if`'s testing `E`, nested according to the structure of `d`.)

14. Page 44, l. -3: min(|d|, e|) ⇒ min(|d|, |e|)

15. Page 45, rest of exercise 2.6: all wrong, the weight $w(\mathtt{d})$ doesn't decrease for some `d`'s! Solution (thanks Nils!): Define w as follows, using auxiliary r:

$$\begin{array}{lcll} w(\mathtt{d}) & = & |\mathtt{d}| - r(\mathtt{d}) & \text{where} \\ r(a) & = & 1 & \text{for any atom } \mathtt{a} \\ r((\mathtt{d}_1.\mathtt{d}_2)) & = & 1 + r(\mathtt{d}_2) & r(\mathtt{d}) = \text{length of right spine of } \mathtt{d} \end{array}$$

    Exercise: Using this w, find a running time bound on the equality testing program of Section 2.4.

Chapter 3

1. Page 48, **Definition 3.1.2**... Language `M` *can simulate* language `L` if ...

2. Just below: Equivalently, `M` can simulate `L` iff ...

3. Footnote, page 48: Recall that `Vars` = $\{\mathtt{V}_0, \mathtt{V}_1, \ldots\}$.

4. Page 49, line -9: delete third `(`.

5. Page 50, **Definition 3.3.1**: delete " = `L`-*data* ".

6. Page 51, ↑ below diagram: change to ∈.

7. Page 53, line 2: delete $f(\mathtt{p}) \in \mathtt{T} - programs$.

8. Page 53, paragraph above **3.4.1**: move to **Definition 3.1.1**.

9. Page 54, title of **3.4.2**: "interpretation".

10. Page 54, line -3: omit semicolons.

11. Page 55, line -2: omit `GOTO`.

12. Page 56, second program, line -3: `Replace`.

13. Page 59, **Theorem 3.6.2**: Language `M` can simulate language `L` if ...

14. Page 61, **Figure 3.4**, line 3: `while Y do` ...

15. Page 62, **Figure 3.5**, line 2: `A := hd (tl A);`
16. Page 63, middle: for atom $\mathtt{a}_i$, `v` is the list $\mathtt{nil}^{i-2}$ of $i-2$ `nil`'s.
17. Page 64, **Exercise 3.1**: replace `T`-*programs* by `L`-*programs*.

Chapter 4

1. Page 69, **Proposition 4.1.1**: replace `WHILE`-*programs* by $\mathtt{WHILE}^{\mathtt{1var}}$-*programs*.
2. Page 69, line -6: change to `C := hd (tl P);`
3. Page 69, line -5: change to `Cd := (C.nil);`
4. Page 70, **Figure 4.1**, line 9: `cons T U;`
5. Page 70, **Figure 4.1**, line 11: `=? T U;`
6. Page 70, line -5: "Then" ⇒ "Suppose"
7. Page 72, **Figure 4.2**, line 6: remove `)`
8. Page 71, **Definition 4.2.2**: *regarded as an* `I`-*program* ⇒ *regarded as* `I`-*data*
9. Page 74, **Exercise 4.1**: Prove that for all 1-variable...
10. Page 74, **Exercise 4.1** and **Exercise 4.2**: `i1var` ⇒ `u1var`
11. Page 74, **Exercise 4.3**: last `s` ⇒ `d`.

Chapter 5

In this chapter $\mathbb{D}$ stands for $\mathbb{D}_A$ for a fixed A such that

$$A \supseteq \{:=, ;, \mathtt{while}, \mathtt{var}, \mathtt{quote}, \mathtt{cons}, \mathtt{hd}, \mathtt{tl}, =?, \mathtt{nil}\}$$

.

1. Page 76, **Theorem 5.2.1**, line 2: and for all `s` ⇒ and for all `d`
2. Page 76, line -2: `ConsExp`.
3. Page 78, line 8: missing "`;`".
4. Page 78, line 12: `else` branch ⇒ `write` command.
5. Page 79, **Theorem 5.4.2**, 2 lines above: $= \Rightarrow \simeq$.
6. Page 79, **Theorem 5.4.2**: *Proof.* Assume A is extensional, and both ....
7. Page 80, line -8: • Is the set $\{\mathtt{d} \mid [\![\mathtt{p}]\!](\mathtt{d}) \text{ converges}\}$ a finite set? Infinite?

8. Page 81, line 11: missing `write X`.
9. Page 82, line 11: omit `'`.
10. Page 83 **Lemma 5.7.1** *Proof.*: For simplicity assume the only atom is `nil`. (The idea is easily extended to any finite atom set.)
11. Page 84, line 13: as needed for 2 ⇒ sufficient for 2.
12. Page 85, line 1: then 3 ⇒ then 1.
13. Page 85, line 10: range of A ⇒ range of f.
14. Page 85, program line 3: `j` ⇒ `nil`.
15. Page 85, line -2,3: only `WHILE` and `I` languages have been.
16. Page 86, line 1: A set A is *recursive* (also called *decidable*) iff there is a program `p` that decides the problem $x \in A$?, and terminates on all inputs.
17. Page 86, **Exercise 5.2**: Define a `WHILE`-computable and total function g satisfying $g(\mathtt{p}) = \mathtt{not}\ [\![\mathtt{p}]\!]^{\mathtt{FL}}(\mathtt{p})$ any `WHILE-forloop`-program `p`. Prove that g is not computable by any `WHILE-forloop`-program.

    Consequence: the `WHILE` language cannot be simulated by the `WHILE-forloop` language.
18. Page 87, **Exercise 5.14**: Let $I\!\!D$ be ordered as in **Lemma 5.7.1**. Show that an infinite set A can be enumerated in increasing order (i.e., is the range of an increasing function) if and only if it is decidable.
19. Page 87, **Exercise 5.15**: while ⇒ `WHILE`
20. Page 87, **Exercise 5.17**: Show that there exists a *fixed* program $\mathtt{p}_0$ such that determination of whether $[\![\mathtt{p}_0]\!](\mathtt{d})$ terminates for $\mathtt{d} \in I\!\!D$ is undecidable.

Chapter 6

1. Page 90, line -7: how fast can an interpreter be
2. Page 94, line 3: `p`, we have $[\![[\![\ldots$
3. Page 99, lines 7, 8, 9:

    2. Interpretation versus specialization plus execution:

    $$time_{\mathtt{int}}(\mathtt{p.d}) \text{ versus } time_{\mathtt{spec}}(\mathtt{int.p}) + time_{\mathtt{int_p}}(\mathtt{d})$$

    If program `p` is to be run just once...
4. Page 99, line 12: ison 1 is more fair since ...

5. Page 99, line -5: `(int.source)` ⇒ `(spec.int)`

6. Page 100, line -4: `in` ⇒ `input`

7. Page 102, line -2: `while` loop ⇒ `if` statement

8. Page 105, **Figure 6.3**, line 3: $\underline{\texttt{a}}\texttt{(m-1,}\underline{\texttt{a}}\texttt{(m,}\underline{\texttt{n-1))}}$

9. Page 106, line -7: prameters ⇒ parameters

10. Page 109, **Exercise 6.1**, line 2: target form `t`. Line 3: omit last "the".

11. Page 109, **Exercise 6.3**: bootsrapping ⇒ bootstrapping

Chapter 7

1. Page 111 middle: Church-Turing thesis: that *all reasonable computation models are equivalent.*

2. Page 112, line -9: omit "."

3. Page 112, line -1: stores ⇒ states

4. Page 113, **Definition 7.2.1**: Definition 9.1.1 ⇒ Definition 2.1.1

5. Page 113, line -1: $\texttt{I}'_\ell \Rightarrow \texttt{I}_{\ell'}$

6. Page 114, program: remove `I` from all instruction labels

7. **Figure 7.1**: add $\texttt{p} \vdash$ at start of each line; and replace every $\texttt{I}\ell$ by $\texttt{I}_\ell$

8. Page 116 line -1: `X0`,`X1`,...,`Xm`

9. **Figure 7.3**, line 3: $\texttt{S}'\underline{\texttt{S}} \Rightarrow \underline{\texttt{S}'}\texttt{S}$

10. **Figure 7.3**, line -2: If $\texttt{I}_\ell =$ " `if S goto` ℓ' " ⇒
    If $\texttt{I}_\ell =$ " `if S goto` ℓ' `else` ℓ'' "

11. **Figure 7.3**, line -1: $\ell' \Rightarrow \ell''$

12. **Figure 7.4**, line 3: omit last $'$

13. **Section 7.4**: add at end:

    Language `2CM` is identical to language `CM`, except that every program has exactly two counters.

14. Page 119, line -6: `SRAM` ⇒ `RAM`

15. Page 122, line -6: $m \Rightarrow n$ (twice)

16. Page 122, line -5: $C_1 \Rightarrow C_0$ (twice)

17. Page 122, line -5: add at end: and $C_0 \leadsto C_1, \ldots, C_{n-1} \leadsto C_n$.

18. Page 123, lines 1,2,3: $\Sigma \Rightarrow \Sigma^*$

19. Page 124, **Exercise 7.5**: Assume x, y are initial values of two different counters.

20. Page 124, **Exercise 7.6**: more than memory cell $\Rightarrow$ more than one memory cell

Chapter 8

1. Page 129 bottom: omit last sentence.

2. **Figure 8.4**: $c_{\mathbb{D}} \Rightarrow bin$

3. **Figure 8.5, line 3**: omit 1+. Omit line 4.

4. **Section 8.3.2**: omit all except first sentence, defining *bin*

5. **Section 8.4**, line 2: omit first "the"

6. **Section 8.4**, *everywhere*: replace `X0` by `Acc`. Point: to avoidthe redundancy caused by having the value of register 0 on both tape 1 and tape 3.

7. Page 133, line -4: $c_{\texttt{01B}}\tilde{\texttt{L}} \Rightarrow c_{\texttt{01B}}(\tilde{\texttt{L}})$

8. Page 133, line -2: $;\overline{\texttt{I}}_k$

9. Page 134, lines 7, -3: `else` clauses omitted (obvious additions)

10. Page 135, lines 2, 4: $n \Rightarrow k$

11. Page 135, line 4:

$$f(x_1, \ldots, x_k) = y \text{ iff } \texttt{q} \vdash \sigma_0 \to^* \sigma \text{ where } \sigma(0) = y$$

12. Page 135, line 8: `X1` $\Rightarrow$ `X0` (4 places)

13. Page 138, line 6: add `Z := 0` at start of line

14. Page 138, lines 7, 8: $\texttt{Z} \geq 4 \Rightarrow \texttt{Z} \geq 3$

Chapter 9

1. Page 137, line -10: omit `F,`

2. Page 137, line -3: omit "partial"

3. **Figure 9.1**, line -2: `f(E)` $\Rightarrow$ `f E`

4. **Figure 9.1**, line -2: $\mathcal{E}[\![\texttt{B}]\!]\texttt{u = w} \Rightarrow \mathcal{E}[\![\texttt{B}]\!]\texttt{Bu = w}$

5. Page 138, line -8: `(append (cons (tl hd Z) (tl Z)))`
6. **Figure 9.3**: `Cd:=hd hd X;` ⇒ `Cd := cons (hd hd X) nil`
7. **Figure 9.3**, line 6: add `X := hd St;` before `write X`
8. **Figure 9.3**: `var′ X` ⇒ `var′`
9. **Figure 9.3**: `docon′s` ⇒ `docons′`
10. **Figure 9.3**: `cons U T` ⇒ `cons T U`
11. Several places: $\mathbb{D}$ instead of $\mathcal{D}$.
12. Page 146, line 7: omit "and occurrences of `W`"
13. Page 146, middle: `C; C` ⇒ `C; D`
14. Page 146, line -6: omit extra ")"
15. Page 148, line 2: until no redexes occur in the input.
16. Page 148, line 6: if ⇒ for all
17. **Figure 9.5**: this terminates a bit too often.
18. **Exercise 9.1**: Omit third sentence.

Chapter 10

1. Page 151, line 12: interesction ⇒ intersection
2. **Figure 10.1**, line 4: start with `#L`$_{\ell}$ `::=` `#L`$_{\ell'}$
3. **Figure 10.1**, line 7: start with `L`$_{\ell}$`#` `::=` `#L`$_{\ell+1}$
4. **Figure 10.1**, line 9: start with `L`$_{\ell}$`#` `::=` `#L`$_{\ell'}$
5. Page 156, line 6: omit 2 commas
6. Page 156, lines 11,12: wrong!
7. Page 158, line 6: omit $u-$
8. Page 159: the reasoning needs tightening up. Sentence "Further, for every…" isn't clear. Also, $\vec{i}$ contains #, which isn't an index.
9. Page 160: needs some reworking.
10. Page 161, lines 15, 16: $u_i \quad\Rightarrow\quad v_i$
11. Page 162, line 10: $u_{i_1}u_{i_2}\dots u_{i_m} = v_{i_1}v_{i_2}\dots v_{i_m}$
12. **Exercise 10.3**, line 2: $(u_n, v_n) \quad\Rightarrow\quad (u_k, v_k)$

Chapter 11

1. Page 167, quote line 4: rational integers $\Rightarrow$ rational numbers
2. **Exercise 11.4**: Use the binomial theorem to prove that for all $n \in I\!N$ and all $k \in \{0, \dots, n\}$

$$\binom{n}{k} \leq 2^n$$

(Not so hard, no hint necessary!)

Chapter 12

Sections 12.1, 12.2, 12.4.1 will be given a different and clearer presentation in the lectures.

1. Page 194, line -6: h is total recursive, $\Rightarrow$ h is total, and computable since f, g are computable,
2. Page 194, line -3: Define $g(\mathtt{d}) = f(tl(\mathtt{d}))$ $\Rightarrow$ Define $g(\mathtt{d}) = tl(f(\mathtt{d}))$
3. Page 195, lines -3, -4:
   2. Each inference rule R_r has a *type* $P_1 \times \ldots \times P_k \rightarrow P$ where ...
4. Page 197, lines -1,-2, page 198, lines 1,2:

   Define a *proof tree* t to be a proof tree form such that every subtree

   $$(\mathtt{nil}^r\ \mathtt{d}\ (\mathtt{nil}^{r_1}\ \mathtt{d}_1 \ldots) \ldots (\mathtt{nil}^{r_k}\ \mathtt{d}_k \ldots))$$

   satisfies:

   - Each $\mathtt{d}_i$ is a proof tree; and
   - There is a predicate $R_r \subseteq I\!D^k \times I\!D$ of type $P_1 \times \ldots \times P_k \rightarrow P$ such that

     $$((\mathtt{d}_1, \ldots, \mathtt{d}_k), \mathtt{d}) \in R_r$$

5. Page 199, lines -5, -7: replace F by S
6. Page 201 top: replace by

   Expressions. This is by an easy induction on syntax:

   $$\begin{array}{lcl} \mathtt{F}_{\mathtt{nil}}(d, d') & \equiv & d' = \mathtt{nil} \\ \mathtt{F}_{\mathtt{hd\ E}}(d, d') & \equiv & \exists d''(d = (d'.d'')) \\ \mathtt{F}_{\mathtt{tl\ E}}(d, d') & \equiv & \exists d''(d = (d''.d')) \\ \mathtt{F}_{\mathtt{X}}(d, d') & \equiv & d' = d \\ \mathtt{F}_{(\mathtt{E1.E2})}(d, d') & \equiv & \exists r \exists s\, \mathtt{F}_{\mathtt{E1}}(d, r) \wedge \mathtt{F}_{\mathtt{E2}}(d, s) \wedge d' = (r.s) \end{array}$$

7. **Exercise 12.1**: omit “first”

Chapter 13

1. **Definition 13.2.2**: $h : \mathbb{N}^n \to \mathbb{N}_\perp$

2. Page 207 middle Part 2: interchange m, n

3. Page 209 line 6: $v_0 \quad \Rightarrow \quad v_1$

4. Page 209 displayed equation in item 3:

$$ins_\ell(\overline{s}) = \overline{s'} \text{ iff } \mathtt{I}_\ell : (\ell, s) \to (\ell', s')$$

5. Page 209 line -6: $g(s) = \ldots$

6. Page 210, lines , 11: $\mathbb{N} \quad \Rightarrow \quad \mathbb{D}$

7. Page 210, last line of program: `write Y` $\Rightarrow$ `write New`

8. Page 210, line -3, -2: There exist `WHILE`-computable total functions $U : \mathbb{D} \to \mathbb{D}$ and $T : \mathbb{D} \times \mathbb{D} \times \mathbb{D} \to \mathbb{D}$ such that ...

9. Page 211, line 3: add ")" at end

10. **Exercise 13.5**, paragraph 2: Prove that if f is partial computable, there exists a partial computable function g with $g \approx \varepsilon y \,.\, f(x, y)$. Hint: use dovetailing as in Theorem 5.5.1.

Chapter 14

1. Page 223, line -14: ...hence any $\mathtt{I}^\uparrow$ program can be compiled into `I`.

2. Page 224, **Example 2**: (Remark: The program uses a version of $\mathtt{I}^\uparrow$ with sysntax extended as seen in Chapter 2.)

Chapter 15

1. Page 241, line 1: program $\Rightarrow$ problem

2. Page 242, line -13: that that $\Rightarrow$ that

Chapter 16

1. Page 241, line 1: program $\Rightarrow$ problem

2. Page 242, line -13: that that $\Rightarrow$ that

3. Page 249 bottom:

   - In `WHILE`, `GOTO` and `F`, the only atom used is `nil`.
   - A *fixed input set*, namely $\{0, 1\}^*$ or a subset ...

- In RAM, the input is offline rather than in a register. All registers are initialized to 0, except that register R0 is initialized to the length of the input.

4. Page 252, lines 1, 5 and -4: L-program q $\Rightarrow$ M-program q
5. **Definition 16.4.2**: missing one clause

$$\mathcal{T}[\![\texttt{atom=?}\quad \texttt{E}]\!]\sigma = 1 + \mathcal{T}[\![\texttt{E}]\!]\sigma$$

6. Page 254, line -4: should be

$$\texttt{C} \vdash \sigma \rightarrow \sigma', \text{ and } \texttt{C;while E do C} \vdash^{time} \sigma' \Rightarrow t'$$

7. Page 258, line -3: SRAMro $\Rightarrow$ SRAM

Chapter 17

1. **Figure 17.3**, line 1: wWhere $\Rightarrow$ Where
2. Page 267, line 1: 1 $\Rightarrow$ 0
3. Page 268, line 9: if $\texttt{a}_i = 0$, else $\Rightarrow$ if $\texttt{d}_i = 0$, else
4. **Figure 17.5**, time $t = 8$: $\texttt{Hd}_8$ should be 1
5. **Exercise 17.1**: For input-output, let the `readin` program have two special vaiables: `eof`, which has value `true` if no more input is left to be read; and `next` which, whenever referenced, yields the next input value if any exists. If there is no remaining input, then execution aborts.

Chapter 18

1. Page 271, line 3: omit "invariant"
2. Page 271, lines 9-11: ...(e.g., the choice to represent ...adjacency lists should not make a complexity difference)
3. **Lemma 18.1.3** end: $\text{LINTIME}^{\texttt{L}} = \text{LINTIME}^{\texttt{M}}$, and analogously for PTIME under relations $\preceq^{ptime}, \equiv^{ptime}$.
4. Page 274, middle: change to

   Since an SRAM-program can at most increase any cell Xi by 1 in one step, none of the values v_i can exceed $t + n$ in value (since the initial value of every cell Xi is 0, excepting X0 which is initialized to n.)

5. Page 274, lines -7, -6: change to

   ...takes time more than $time_{\texttt{q}}^{\texttt{TM}}(\texttt{d}) \leq a \cdot u \log u$ to simulate, where $u = time_{\texttt{p}}^{\texttt{SRAM}}(\texttt{d}) + n$. Thus one simulation ...

6. **Theorem 18.2.4**:

   $\text{PTIME}^{\texttt{TM}} = \text{PTIME}^{\texttt{GOTO}} = \text{PTIME}^{\texttt{SRAM}} = \text{PTIME}^{\texttt{WHILE}} = \text{PTIME}^{\texttt{I}}$

7. Page 276 bottom:

   $\text{LINTIME}^{\texttt{TM}} = \text{LINTIME}^{\texttt{GOTO}} = \text{LINTIME}^{\texttt{SRAM}} = \text{LINTIME}^{\texttt{WHILE}} = \text{LINTIME}^{\texttt{I}}$

8. Page 279, lines -10 to –8:

   We shall prove that if

   $$M = (\Sigma, Q, \ell_{init}, \ell_{fin}, T)$$

   is a 1-tape Turing machine running in time f and $\varepsilon > 0$, then there is a 2-tape machine

   $$M' = (\Sigma', Q', \ell'_{init}, \ell'_{fin}, T')$$

9. **Figure 18.1**: remove extra commas in lines (8), (9)

10. **Exercise 18.1**: program-independent constant-factor slowdown

11. **Exercise 18.5**: with at most a constant-factor slowdown. Is the slowdown program-dependent?

Chapter 19

1. Page 288, line -2:

   $$time_{\texttt{tu}}(\texttt{p.d.nil}^n) \le k \cdot \min(n, time_{\texttt{p}}(\texttt{d}))$$

2. Page 289, lines -4, -5:

   $$time_{\texttt{tu}}(\texttt{p.d.nil}^n) \le k \cdot time_{\texttt{p}}(\texttt{d})$$

   $$time_{\texttt{tu}}(\texttt{p.d.nil}^n) \le k \cdot n$$

3. Page 292, program line 2:

   ```
   Timebound := [[b]](X): (*Insert body of b here *)
   ```

4. **Definition 19.5.2**: ...and a constant $c > 0$ such that ...

5. **Definition 19.5.2** line 3: ... $time_{\texttt{b}}(\texttt{nil}^n) \le c \cdot f(n)$

6. **Exercise 19.5**: This exercise requires a model not yet introduced: the *read-only* variants of Section 21.2, page 316.

7. **Exercise 19.2**: Prove Theorem 19.5.4 for a special case: that there are problems ...

Chapter 20

1. Page 305, line 4: omit "program"
2. Page 305, line -5: ... a function f such that $f = [\![\mathtt{p}]\!]$ implies $\mathtt{p} \notin Q$, ...
3. Page 306, line 3 after figure: different from $[\![\mathtt{p}_k]\!](n)$ where ...

Chapter 21

1. Page 317 line -8: omit " $= \mathtt{a}_1\ \mathtt{a}_2 \ldots \mathtt{a}_n$ "
2. **Definition 21.1.10**: Extra) in 3 superscripts
3. Page 322 line 9: should start with $(\ell, \ldots \mathtt{B}_1 \mathtt{L}_1 \underline{\mathtt{S}_1} \mathtt{B} \ldots$
4. **Proposition 21.6.1** item 1: $x \cdot y \quad \Rightarrow \quad x - y$
5. **Exercise 21.2**: Prove Corollary 21.1.6.

Chapter 22

1. **Exercise 22.1**: add *Hint:* "if" is immediate from earlier results since nondeterministic Turing machines include deterministic ones. For "only if" modify the pattern of Theorem 13.4.1 to apply to a given nondeterministic Turing machine.

Chapter 23

1. **Figure 23.2** line 6: `Counter := n;`
2. **Figure 23.4** line 11: `for k := 1 to r do {`
3. **Theorem 23.4.3**: $\text{NSPACE}(f) \subseteq \bigcup_b \text{SPACE}(b \cdot f^2)$
4. **Exercise 23.2**: Estimate the running time of the state transition-searching algorithm of Theorem 23.3.4.
5. **Exercise 23.4**: Estimate the running time of the LOGSPACE algorithm of Theorem 23.3.2 for deciding membership in $\overline{\text{GAP}}$.

Chapter 24

1. **Exercise 24.3**: Replace `GOTO` by `GOTOro`, and `Fro` by `F+ro`.

Chapter 25

1. Page 365, line -7: ... may have many hardest problems.
2. Page 367, line -8:
   GAP = $\{(G, v_0, v_{end}) \mid$ directed graph $G = (V, E)$ has a path from vertex v_0 to v_{end} $\}$
3. **Proposition 25.3.5**: If A is $\leq$-hard for $\mathcal{D}$, and $A \leq B$, and $B \in \mathcal{D}$, then B is also hard for $\mathcal{D}$.
4. Page 375, lines 15-17: However for this approach to be useful it is necessary that problem H be well-chosen: simpy stated, and such that it can be reduced to many interesting problems.
5. **Definition 25.6.1**: ... $\mathit{time}_{\mathtt{p}}(\mathtt{d}) \leq a \cdot |\mathtt{d}|$ for all $\mathtt{d} \in \mathbb{D}$.
6. **Theorem 25.6.6**: ... `p` is a `while`-free `I`-program and `p` accepts `d`.

Chapter 26

1. Page 384, line 2: ... and adding $i + 1$ for each
2. **Lemma 26.1.5**, line 3: remove "`p` and"
3. Page 385, lines 18, 20 (definitions of `H, T`): `true` iff 1 is the ith bit of ...
4. Page 394, line 18: ... any won position $p \in W$ is

Chapter 27

1. Page 403, line -11: VertexCover
2. Page 403, line -4: $S \setminus C \quad \Rightarrow \quad V \setminus C$
3. Page 404, line -10: $(u, w) \in E$

Appendix section A.3.11

1. Item 1: $g(n) \leq r \cdot f(n)$
2. Item 2: for all but finitely many n, $g(n) \geq r \cdot f(n)$