

Solutions to Selected Exercises for Chapter 4 of R. Gregory Taylor, *Models of Computation and Formal Languages* (Oxford University Press: New York, 1998)

© Oxford University Press

Solutions to Exercises for § 4.1

- 4.1.2 (a) true
(b) true
(c) false

4.1.3.

$$\begin{aligned} & \& \rightarrow \cdot \varepsilon \\ & \dots \\ & \alpha_1 \rightarrow \& \beta_1 \\ & \dots \\ & \alpha_2 \rightarrow \& \beta_2 \\ & \dots \end{aligned}$$

The single terminal production here must, in general, be the very first production. Were this not the case, some other production application could intervene so as to alter a substituted β_1 or β_2 .

Solutions to Exercises for § 4.2

4.2.2 (a)

$$\begin{aligned} & @ \alpha \rightarrow a @ \text{ for } \alpha \in \Sigma \\ & * a @ \rightarrow \cdot I \\ & \$ a @ \rightarrow \cdot I \\ & * a \rightarrow * \\ & * b \rightarrow \$ \\ & \$ b \rightarrow \$ \\ & \varepsilon \rightarrow * @ \end{aligned}$$

(d)

$$\begin{aligned} & * a \rightarrow a * \\ & * b \rightarrow \$ \\ & a \$ a \rightarrow \$ \\ & \$ \rightarrow \cdot I \\ & \varepsilon \rightarrow * \end{aligned}$$

An alternative solution is

$$\begin{aligned} & b \rightarrow @ \\ & a @ a \rightarrow @ \\ & @ \rightarrow \cdot I \end{aligned}$$

(e)

$\rho\alpha \rightarrow a\rho$ for $\alpha \in \Sigma$
 $\rho_a\alpha \rightarrow a\rho_a$ for $\alpha \in \Sigma$
 $\rho_b\alpha \rightarrow a\rho_b$ for $\alpha \in \Sigma$
 $a\rho_a\rho \rightarrow \rho$
 $b\rho_b\rho \rightarrow \rho$
 $\lambda a \rightarrow \lambda\rho_a$
 $\lambda b \rightarrow \lambda\rho_b$
 $\lambda\rho \rightarrow .I$
 $\lambda\rho_a\rho \rightarrow .I$
 $\lambda\rho_b\rho \rightarrow .I$
 $\varepsilon \rightarrow \lambda\rho$

4.2.3. $\{(ab)^n b^m \mid n \geq 0, m \geq 0\}$

4.2.4. Click twice on icon **Exercise 4.2.4** in the accompanying software.

- 4.2.5. (a) 17
(b) 26
(c) 37
(d) 50
(e) $(n/2)^2 + n + 2$

Solutions to Exercises for § 4.3

4.3.1 4
 $\langle 4, 7, 1 \rangle$

4.3.3 (a) $IIII$
(b) I
(c) projection function p_2^2

4.3.5 (a) $III \rightarrow II$

(b) $III \rightarrow II$
 $II \rightarrow .I$
 $I \rightarrow .II$

(c) $I*I \rightarrow \varepsilon$
 $* \rightarrow I$

4.3.6. Suppose that Markov algorithm schema S is given, and let Π be the production sequence of S . If Π happens to contain any productions with left-hand side ε , then it is already true that S can halt only as the result of applying a terminal production. Otherwise, add the production

$$\varepsilon \rightarrow .\varepsilon$$

at the very end of production sequence Π . This change will have no effect upon M 's behavior qua transducer. However, it will now be the case that S halts only by applying this or some other terminal production.

Solutions to Exercises for § 4.4

- 4.4.1 (a) Let S be a labeled Markov algorithm schema whose very first production is π . Now π may or may not be a labeled production. If it is labeled, then let us assume that its label is \mathcal{L} . On the other hand, if it is not labeled, let \mathcal{L} be its new label. Now, for every production of S that lacks a goto, let \mathcal{L} be the new goto of that production. The algorithm schema S' that results from S by making these changes differs from S in no way that will affect algorithm execution. Moreover, every production of S' has a goto.
 (b) By (a) we may assume without loss of generality that absolutely every production of S has a goto. Now form S' by adding labeled terminal production

$$\mathcal{L}: \varepsilon \rightarrow \cdot \varepsilon$$

at the very bottom of Π , where \mathcal{L} is a label not occurring within Π . Clearly, S' does not differ operationally from S since the added production causes no change whatever to any computation word and is applicable only if no production of S is applicable.

- 4.4.2. We said that application of a production with meaningless goto would cause an algorithm to halt execution. Consequently, any production with meaningless goto can be replaced by the corresponding terminal production without goto. The result of making all such changes is a new algorithm schema S' that is equivalent to S .

Solutions to Exercises for § 4.5

4.5.1. $\lceil 11*111 \rceil = 222122_3 = 2 \cdot 3^5 + 2 \cdot 3^4 + 2 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 2 \cdot 3^0 = 486 + 162 + 54 + 9 + 6 + 2 = 719$

4.5.2. $encode_arg(1,2) = 719$

4.5.3.

$$prod_lhs(i) = \begin{cases} \lceil *1 \rceil = 7 & \text{if } i=1 \\ \lceil * \rceil = 1 & \text{if } i=2 \\ 0 & \text{otherwise} \end{cases}$$

$$prod_rhs(i) = \begin{cases} \lceil * \rceil = 1 & \text{if } i=1 \\ \lceil \varepsilon \rceil = 0 & \text{if } i=2 \\ 0 & \text{otherwise} \end{cases}$$

4.5.4. $length(719) = length(\lceil 11*111 \rceil) = 6$ since $3^5 = 243 < 719 < 729 = 3^6$

4.5.5. $segment(3,4,719) = (719 \div 9) \bmod 9 = 7$, which just happens to be $\lceil *1 \rceil$

4.5.6. Either $n=7$ or $n=17$ will work since both $*1$ and $11*$ occur within $11*111$. Other possibilities exist as well.

4.5.7. $leftmost_occur_start_position(7,719) = 3$ since $*1$ occurs within $11*111$ starting in the third position.

4.5.8. $subst(1,3,4,719) = (719 \bmod 3^2) + (1 \cdot 3^2) + (719 \div 3^4) \cdot 3^{3+1-1}$
 $= (719 \bmod 9) + (1 \cdot 9) + (719 \div 81) \cdot 27$
 $= 8 + 9 + 8 \cdot 27$
 $= 233$

which we can verify independently by noting that string $11*11$ with $\lceil 11*11 \rceil = 233$ is the result of substituting string $*$ with $\lceil * \rceil = 1$ for the string extending from position 3 to position 4 inclusive within string $11*111$ where $\lceil 11*111 \rceil = 719$.

4.5.9. $first_applicable_rule(53) = 2$

4.5.10. $apply_rule(2,53) = 26$, which just happens to be $\lceil 111 \rceil$

4.5.11. $\chi_{term_rule}(1)=0$ and $\chi_{term_rule}(2)=1$

- 4.5.12. (a) $step(6533,0)=\langle 6533,1 \rangle$
 (b) $step(6533,1)=\langle 2159,1 \rangle$
 (c) $step(6533,2)=\langle 701,1 \rangle$
 (d) $step(6533,3)=\langle 215,1 \rangle$
 (e) $step(6533,4)=\langle 53,2 \rangle$
 (f) $step(6533,5)=\langle 26,0 \rangle$
 (g) $step(6533,6)=\langle 26,0 \rangle$

4.5.13. We have

$$\begin{aligned} & length[step(encode_arg(2,3),length_comp(encode_arg(2,3)))_1] \div 1 \\ &= length[step(6533,length_comp(6533))_1 \div 1] \\ &= length[step(6533,5)_1] \div 1 \\ &= length(26) \div 1 \\ &= 3 \div 1 \\ &= 2 \end{aligned}$$

which is consistent with the fact that $f(2,3)=p_1^2(2,3)=2$.

4.5.17. Suppose that S halts execution for some input words representing k -tuples of arguments for which f is in fact undefined. By Definition 4.8 this can only mean that, started upon certain input words, S produces an output word containing symbols other than l . So the idea behind the solution that follows is that, before halting, S will now check whether any work alphabet symbols other than l s remain in the current computation word and, if so, will enter an infinite loop.

First, by Exercises 4.1.3 and 4.3.6 together, we may assume that S halts as the result of applying some unique terminal production

$$\alpha \rightarrow \beta$$

We replace this production by the production with goto

$$\alpha \rightarrow \beta; \mathcal{L}$$

and also add, at the very bottom of S' production sequence, the new labeled instruction

$$\mathcal{L}: \quad \varepsilon \rightarrow \$$$

where $\$$ is a new symbol not occurring in S' work alphabet Γ . Further, we add the production

$$\$l \rightarrow l\$$$

at the very *beginning* of S' production sequence. Immediately following that new, first production, we add, for every symbol σ in Γ other than l , the production

$$\$ \sigma \rightarrow \$ \sigma$$

And following that group of productions, we add the single terminal production

$$\$ \rightarrow \varepsilon$$

Solutions to Exercises for § 4.6

- 4.6.1. We supply the following hint. Applying a production may increase the length of the current computation word. However, the amount by which it may be increased is $O(1)$ worst case. Why?
- 4.6.2 (a) This demonstration is trivial since any Markov algorithm schema *without* labels is, by definition, a Markov algorithm schema *with* labels.
- (b) Referring to the Markov algorithm schema without labels constructed within the proof of Theorem 4.1, note that each application of labeled production $\alpha_j \rightarrow \beta_j$; L_j of S_{Lab} will be implemented *directly* within S_{Lab} 's computation by applying production ${}_j\alpha_j \rightarrow \equiv_j \beta_j$. However, preceding that application, productions of the form ${}_j\alpha \rightarrow \alpha_j$ will have been applied $O(space_{S_{Lab}}(n))$ times—and, hence, $O(time_{S_{Lab}}(n))$ times by Exercise 4.6.1. Similarly, following the application of production ${}_j\alpha_j \rightarrow \equiv_j \beta_j$, there will be $O(time_{S_{Lab}}(n))$ applications of productions of the form $\alpha \equiv_j \rightarrow \equiv_j \alpha$. Very briefly, each application of a labeled production of S_{Lab} will be implemented overall by $O(time_{S_{Lab}}(n))$ applications of productions of S . But, by definition, there are $O(time_{S_{Lab}}(n))$ such production applications to be implemented, from which it follows that S computes in $O([time_{S_{Lab}}(n)]^2)$ steps.
- (c) Suppose that labeled Markov algorithm schema S_{Lab} accepts language L in $O(p(n))$ steps for some polynomial function $p(n) \geq n$ for sufficiently large n . Then, by (b), there exists a Markov algorithm schema S without labels that accepts L in $O([p(n)]^2)$ steps. But, of course, $[p(n)]^2$ is itself a polynomial function in n . The other direction is similar but uses (a).

4.6.3. First, recall that $time_M(n)$, as defined in Definition 1.8, is a certain maximum taken over all input words w with $|w|=n$. Hence, although there do exist words w for which the number of applications of Group 2 productions will be exactly $time_M(n)$, this will not be true in general. And at (2), we are talking about an *arbitrary* input word w with $|w|=n$.

Solutions to Exercises for § 4.7

- 4.7.7. (a) For the purposes of encoding, a Markov algorithm schema may be taken to be an ordered sequence of productions. (In other words, we are presently identifying Markov algorithm schema $S = \langle \Sigma, \Gamma, \Pi \rangle$ with its production sequence Π .) Our *Markov algorithm schema description alphabet* Ψ will contain symbols I , \rightarrow , $.$, $,$, $,$, and $;$. In addition, Ψ might also contain ϵ as a representation of the empty word. In order to represent any member of an enumerable set of auxiliary symbols, we use α followed by zero or more occurrences of symbol \prime . Codes are assigned to members of Ψ in accordance with the following table.

SYMBOL	CODE	SYMBOL	CODE
α	1	ϵ	5
I	2	\rightarrow	6
$.$	3	$,$	7
\rightarrow	4	$;$	8

Productions sequences are now assigned codes by raising primes to powers in accordance with the Euler-Gödel scheme described in § 2.4. To take a very simple example, the single-member production sequence

$$I \rightarrow .II$$

which happens to compute the successor function $f(n)=n+1$, will be encoded as

$$2^{\ulcorner I \urcorner} \cdot 3^{\ulcorner \rightarrow \urcorner} \cdot 5^{\ulcorner . \urcorner} \cdot 7^{\ulcorner I \urcorner} \cdot 11^{\ulcorner II \urcorner}$$

$$= 2^3 \cdot 3^4 \cdot 5^5 \cdot 7^3 \cdot 11^3$$

In the case of production sequences with several productions, symbol ; can be used to separate productions. We have hereby associated a unique natural number with each Markov algorithm schema having input alphabet $\Sigma=\{1\}$. Accordingly, there exists a 1-1 correspondence between this family of Markov algorithm schemata and a certain (proper) subset of \mathcal{N} . We conclude that this family of Markov algorithm schemata is countable.

(b) It is possible to regard every Markov algorithm schema S with $\Sigma=\{1\}$ as computing some unary partial number-theoretic function f_S (cf. Remark 1.5.1 and the ensuing discussion). Moreover, as described in the solution to (a) above, each such S can be encoded as a natural number $\lceil S \rceil$. As a further simplification, we mention that it is possible to revise our encoding scheme in such a way that every natural number becomes the code of a unique Markov algorithm schema. (Cf. the discussion of § 2.4.) We sketch the operation of a universal Markov algorithm S_U on any input word of the form

$$1^{n+1} * 1^{m+1}$$

representing natural numbers n and m . S_U first decodes n so as to obtain the production sequence of that unique Markov algorithm schema S with $\lceil S \rceil = n$. We may imagine that the result will be some more straightforward representation of the production sequence of S within S_U 's current computation word, the last part of which is yet $*1^{m+1}$. Next, S_U proceeds to simulate S 's computation for input word 1^{m+1} . If $f_S(m)$ happens to be defined, then S_U 's output word will be $1^{f_S(m)+1}$. Otherwise, we may assume without loss of generality that S_U never halts.