Appendix A
Results and Proofs Omitted in the Text

We begin with the assertion at the beginning of Chapter 1, that a type 1 language can be generated by a grammar whose productions are context-sensitive.

Note that, at this point in Chapter 1, $S \longrightarrow \varepsilon$ is not allowed as a production in a type 1 grammar. However Lemma A.2 below is true if modified to allow it, adding "except possibly $S \longrightarrow \varepsilon$". This is because the arguments in Lemma 1.1 and Cor. 1.2 apply.

Lemma A.1. *If $G = (V_N, V_T, P, S)$ is a grammar of type 0 or 1, then $L_G = L_{G'}$ for some grammar G' of the same type, such that all productions of G' are either of the form $\alpha \longrightarrow \beta$, where α, β are strings of non-terminal symbols, or of the form $A \to a$, where A is a non-terminal symbol and a is a terminal symbol.*

Proof. For every $a \in V_T$, take a new letter X_a. Let $G' = (V'_N, V_T, P', S)$, where $V'_N = V_N \cup \{X_a \mid a \in V_T\}$ and P' consists of:

	$X_a \longrightarrow a$	for $a \in V_T$
and	$\alpha' \longrightarrow \beta'$	for $\alpha \longrightarrow \beta$ in P, where α' and β' are obtained from α, β by replacing every occurrence of a letter $a \in V_T$ by X_a.

Then G' is of the same type as G, and is the required grammar. For if $\gamma \in L_G$, modify a G-derivation of γ from S, by replacing every production $\alpha \longrightarrow \beta$ used by $\alpha' \longrightarrow \beta'$, to obtain a G'-derivation of γ'. Then by use of the productions $X_a \longrightarrow a$, we obtain a G'-derivation of γ. Hence $L_G \subseteq L_{G'}$.

Conversely, given a G'-derivation of $\gamma \in L_{G'}$, when a production $X_a \longrightarrow a$ is used, the resulting occurrence of a is never changed. Move this production to the end of the derivation, replacing the occurrence of a by X_a until the production $X_a \longrightarrow a$ is used. Repeating this procedure, we obtain a G'-derivation of γ in which all uses of productions $\alpha' \longrightarrow \beta'$ occur first. This produces a word in the new letters X_a which is then converted to γ, so this word must be γ'. Now replace every use of a production $\alpha' \longrightarrow \beta'$ by the production $\alpha \longrightarrow \beta$ and delete all uses of productions $X_a \longrightarrow a$ at the end. This gives a G-derivation of γ from S. Hence $L_G = L_{G'}$. □

Lemma A.2. *Let G be a type 1 grammar. Then $L_G = L_{G'}$ for some grammar G' in which all productions are context-sensitive.*

Proof. We can assume the productions of G are as in Lemma A.1. Given a production $a_1 \dots a_n \longrightarrow b_1 \dots b_m$ ($m \geq n$, a_i, b_i non-terminal letters) which is not context-sensitive (so $n > 1$), modify G as follows. Add new non-terminal letters $A_1, \dots A_n$ and $B_1, \dots B_m$ (distinct, even if the a_i, b_i aren't), then replace this production by the productions:

$$
\begin{aligned}
&a_1 \dots a_n \longrightarrow a_1 \dots a_{n-1} A_n && (1)\\
&a_1 \dots a_{n-1} A_n \longrightarrow a_1 \dots a_{n-2} A_{n-1} A_n && (2)\\
&\vdots && \vdots\\
&a_1 A_2 \dots A_n \longrightarrow A_1 \dots A_n && (n)\\
&A_1 \dots A_n \longrightarrow A_1 \dots A_{n-1} B_n \dots B_m && (n+1)\\
&A_1 \dots A_{n-1} B_n \dots B_m \longrightarrow A_1 \dots A_{n-2} B_{n-1} B_n \dots B_m && (n+2)\\
&\vdots && \vdots\\
&A_1 B_2 \dots B_m \longrightarrow B_1 B_2 \dots B_m && (2n)\\
&B_1 B_2 \dots B_m \longrightarrow b_1 B_2 \dots B_m && (2n+1)\\
&b_1 B_2 \dots B_m \longrightarrow b_1 b_2 B_3 \dots B_m && (2n+2)\\
&\vdots && \vdots\\
&b_1 \dots b_{m-1} B_m \longrightarrow b_1 \dots b_m && (2n+m)
\end{aligned}
$$

(The reader should check that these are context-sensitive.)

Call the new grammar G_1. Any use of the old production can be replaced by using these $2n+m$ productions in succession, hence $L_G \subseteq L_{G_1}$. Suppose $\alpha \in L_{G_1}$ and there is a G_1-derivation of α from S (the start symbol) using the new productions. The first time such a production is used, it must be (1), since up to that point none of the new non-terminal letters have appeared. This introduces A_n, and this occurrence of A_n must eventually be changed by use of a production (α is a string of terminal letters). This can only be done by use of (2).

The reason is that the letter to the left of A_n is either from the original alphabet, or the right-hand letter of the right-hand side of a new production, which can only be A_n or B_m. Similarly, the letter to the right of A_n (if any) is either from the original alphabet, or A_1 or B_1. But no word on the left-hand side of the new productions contains any of the words A_nA_n, B_mA_n, A_nA_1 or A_nB_1. Thus (2) is the only production that can be used, so part of the derivation has the form:

$$\dots, u_1 a_1 \dots a_n v_1, u_1 a_1 \dots A_n v_1, \dots, u'_1 a_1 \dots A_n v'_1, u'_1 a_1 \dots A_{n-1} A_n v'_1, \dots$$

which can be replaced by

$$\dots, u_1 a_1 \dots a_n v_1, \dots, u'_1 a_1 \dots a_n v'_1, u'_1 a_1 \dots A_n v'_1, u'_1 a_1 \dots A_{n-1} A_n v'_1, \dots \ .$$

(The use of (1) is moved until just before the use of (2).) Similarly, the next time $A_{n-1}A_n$ is changed by a production, it must be by use of (3), and we can change the derivation so (1), (2) and (3) are used in succession. Eventually we obtain a G_1-derivation in which (1) through (2n+m) are used in succession, to change an occurrence of the string $a_1 \dots a_n$ to $b_1 \dots b_m$. These can be replaced by a single use of the original production.

Continuing, we eventually remove all use of the new productions, giving a G-derivation of α (the fact that the original productions are now used does not affect the argument). Hence $L_{G_1} = L_G$. Finally, repetition of the procedure replacing G by G_1 will remove all productions which are not context-sensitive, giving the required grammar G'. □

The next result to be proved is Lemma 2.18. It is necessary to read the relevant part of Chapter 2 to understand the statement and proof.

Lemma 2.18. There is a primitive recursive function $Next : \mathbb{N} \to \mathbb{N}$ such that

$$Next(Code(c)) = Code(\delta(c))$$

for all $c \in C'$.

Proof. Let $c = (q,a,\alpha,\beta)$, so $Code(c) = 2^q 3^a 5^{\sigma(\alpha)} 7^{\sigma(\beta)}$. Put $x = Code(c)$, and use Lemma 1.10. To simplify notation, we omit subscripts and write R, N, D instead of $R_{T'}$, $N_{T'}$, $D_{T'}$.

(1) If $D(q,a) = 0$, $Code(\delta(c)) = 2^{N(q,a)} 3^{\beta(0)} 5^{\sigma(\alpha')} 7^{\sigma(\beta')}$ and

$$\begin{aligned} N(q,a) &= N(\log_2(x), \log_3(x)) \\ \beta(0) &= \mathrm{rem}(2, \log_7(x)) \\ \sigma(\alpha') &= R(q,a) + 2\alpha(0) + 2^2\alpha(1) + \ldots = R(\log_2(x), \log_3(x)) + 2\log_5(x) \\ \sigma(\beta') &= \beta(1) + 2\beta(2) + \ldots = \mathrm{quo}(2, \sigma(\beta)) = \mathrm{quo}(2, \log_7(x)) \end{aligned}$$

(2) If $D(q,a) = 1$, $Code(\delta(c)) = 2^{N(q,a)} 3^{\alpha(0)} 5^{\sigma(\alpha')} 7^{\sigma(\beta')}$ and similarly

$$\begin{aligned} N(q,a) &= N(\log_2(x), \log_3(x)) \\ \alpha(0) &= \mathrm{rem}(2, \log_5(x)) \\ \sigma(\alpha') &= \mathrm{quo}(2, \log_5(x)) \\ \sigma(\beta') &= R(\log_2(x), \log_3(x)) + 2\log_7(x) \end{aligned}$$

Hence, we define $Next(x) = 2^{F_1(x)} 3^{F_2(x)} 5^{F_3(x)} 7^{F_4(x)}$ (for any $x \in \mathbb{N}$) where, putting $E(x) = D(\log_2(x), \log_3(x))$:

$$\begin{aligned} F_1(x) &= N(\log_2(x), \log_3(x)) \\ F_2(x) &= (1 \dot{-} E(x))\mathrm{rem}(2, \log_7(x)) + E(x)\mathrm{rem}(2, \log_5(x)) \\ F_3(x) &= (1 \dot{-} E(x))\big(R(\log_2(x), \log_3(x)) + 2\log_5(x)\big) + E(x)\mathrm{quo}(2, \log_5(x)) \\ F_4(x) &= (1 \dot{-} E(x)))\mathrm{quo}(2, \log_7(x)) + E(x)\big(R(\log_2(x), \log_3(x)) + 2\log_7(x)\big) \end{aligned}$$

Since $F_1, \ldots, F_4$ are primitive recursive, so is $Next$. □

Now we prove two lemmas on deterministic pushdown automata which are needed at the end of Chapter 4, where these and related ideas are defined. We shall need another definition concerning them.

Definition. A deterministic PDA $M = (Q, F, A, \Gamma, \tau, q_0, z_0)$ is said to *always scan its entire input* if, for all $w \in A^*$, $(q_0, w, z_0) \xrightarrow[M]{} (q, \varepsilon, \gamma)$ for some $q \in Q$ and $\gamma \in \Gamma^*$.

Also, we shall describe a transition of a PDA starting $(q, \varepsilon, \ldots)$ as an ε-transition.

The ways in which M can fail to always scan its entire input are firstly, that it halts without reading the entire word on the tape. This can happen if M empties its stack, or if there is no transition beginning (q, a, z) or (q, ε, z), where M is in state q, a is the next letter on the tape and z is on top of the stack. Secondly, it can happen that M continues indefinitely to use ε-transitions without reading another letter from the tape. This observation is the basis for the construction in the next lemma. This makes use of a state d (the "dead state") to continue to read from the tape when any of the situations above is encountered. There is also an extra final state f to accept any words that M accepts in the second situation, when it continues indefinitely to use ε-transitions. Such a word will be a proper prefix of the word on the tape. (The proper prefixes of a word w are the prefixes of w other than w itself.)

Lemma A.3. *If L is a deterministic language, then $L = L(M')$ for some deterministic PDA M' which always scans its entire input.*

Proof. There is a deterministic PDA $M = (Q, F, A, \Gamma, \tau, q_0, z_0)$ such that $L = L(M)$. There is a new PDA $M' = ((Q \cup \{q_0', d, f\}, F \cup \{f\}, A, \Gamma \cup \{x_0\}, \tau', q_0', x_0)$, where τ' is defined as follows.

(1) $(q_0', \varepsilon, x_0, q_0, z_0 x_0) \in \tau'$.
(2) For all $q \in Q$, $a \in A$ and $z \in \Gamma$, if no transition in τ starts with (q, a, z) or (q, ε, z), then $(q, a, z, d, z) \in \tau'$.
(3) For all $q \in Q$, $a \in A$, $(q, a, x_0, d, x_0) \in \tau'$.
(4) For all $a \in A$, $z \in \Gamma \cup \{x_0\}$, $(d, a, z, d, z) \in \tau'$.
(5) If there is an infinite sequence of configurations of M:

$$(q, \varepsilon, z), (q_1, \varepsilon, \gamma_1), (q_2, \varepsilon, \gamma_2), \ldots$$

where $z \in \Gamma$ and each configuration $(q_i, \varepsilon, \gamma_i)$ is obtained from its predecessor by an ε-transition in τ, then

$$\begin{cases} (q, \varepsilon, z, d, z) \in \tau' & \text{if no } q_i \in F \\ (q, \varepsilon, z, f, z) \in \tau' & \text{if some } q_i \in F \end{cases}$$

(6) For all $z \in \Gamma \cup \{x_0\}$, $(f, \varepsilon, z, d, z) \in \tau'$.
(7) For all $q \in Q$, $a \in A \cup \{\varepsilon\}$ and $z \in \Gamma$, if no transition of τ' starting (q, a, z) has been defined by (2) or (5), and there is a transition $(q, a, z, q', \gamma) \in \tau$, then $(q, a, z, q', \gamma) \in \tau'$.

It is easy to see that M' is deterministic. Note that M', in its initial state, always begins a computation by putting x_0 on the bottom of the stack (using (1)), and this is never erased.

Suppose M' does not always scan its entire input. Then for some $w \in A^*$,

$$(q_0', w, z_0) \xrightarrow[M']{} (q, au, z_1 \dots z_k x_0)$$

where $a \in A$ and au is a suffix of w, $z_i \in \Gamma$, $k \geq 0$, and a is never read. That is, the computation can only be continued by use of ε-transitions. In fact, the computation of M' can be continued using an ε-transition. For if M has no transition beginning (q, ε, z_1), then either by (2) or (7) M' has a transition starting (q, a, z_1), so a can be read from the tape, a contradiction. Thus M has a transition beginning (q, ε, z_1), so either by (5) or (7), M' has a transition starting (q, ε, z_1), as claimed. Repeating this argument, the computation of M can be continued indefinitely using ε-transitions, giving a sequence

$$(q, au, z_1 \dots z_k x_0),\ (q_1, au, \gamma_1 z_2 \dots z_k x_0),\ (q_2, au, \gamma_2 z_2 \dots z_k x_0), \dots$$

Note that q and all q_i are in Q, because no ε transition begins with d, and in state f, the next configuration will be in state d, using (6). Consequently, the ε-transitions used are all transitions of M. Now eventually z_1 must be erased from the stack, that is, some $\gamma_i = \varepsilon$. Otherwise (5) applies to the sequence

$$(q, \varepsilon, z_1),\ (q_1, \varepsilon, \gamma_1),\ (q_2, \varepsilon, \gamma_2), \dots$$

(obtained by using the same transitions used in the sequence above). The first transition used is then given by (5), so q_1 is either d or f, a contradiction.

Similarly, $z_2, \dots, z_k$ are eventually erased, leading to a configuration (q_i, au, x_0). Then for the next move, only a transition in (3) can be used, so $q_{i+1} = d$, a contradiction. Thus M' always scans its entire input.

Finally, we need to show $L(M) = L(M')$. Suppose M accepts $w \in A^*$. There is thus a computation of M beginning with (q_0, w, z_0) which scans all of w and ends in a final state. While a non-empty suffix of w remains on the tape, the transitions used are transitions of M', by (7). This gives a computation of M', using these transitions together with an initial use of the transition in (1), starting in configuration (q_0', w, x_0). If, M is in a final state just after reading all of w, then M' will be in the same state just after reading w, so M' accepts w.

Otherwise, M then uses a sequence of ε-moves until a final state is reached. Either these are transitions of M', so again M' accepts w, or (5) applies and M', just after reading w, enters state f, so accepts w. Thus $L(M) \subseteq L(M')$.

Suppose M does not accept w, and consider a computation of M starting with (q_0, w, z_0). If M halts, then (whether or not all of w has been read), we obtain, using (2) and (3), a corresponding computation of M', starting with (q_0', w, x_0), which enters state d. Since M' is deterministic, it does not accept w. (In state d, only transitions in (4) can be used, and d is not a final state.)

Otherwise, the computation of M can be continued indefinitely, giving a sequence as in (5), where, if all of w has been read, no $q_i \in F$. Up to the point where M stops reading from the tape, there is a corresponding computation of M'. Then by (5), either M' enters state d, or enters state f having read a proper prefix of w. In the latter case, M' then enters state d using an ε-transition from (6). In any case, M' does not accept w. Hence $L(M) = L(M')$. □

We shall need the lemma just proved for the next result, which is used in Chapter 4.

Lemma A.4. *If L is a deterministic language, then $L = L(M')$ for some deterministic PDA M' which has no ε-transitions beginning with a final state.*

Proof. There is a deterministic PDA $M = (Q, F, A, \Gamma, \tau, q_0, z_0)$ such that $L = L(M)$. By Lemma A.3, we can assume M always scans its entire input. Define a new PDA $M' = (Q', F', A, \Gamma, \tau', q_0', z_0)$ as follows:

$$Q' = Q \times \{1,2,3\}$$
$$F' = \{(q,3) \mid q \in Q\}$$
$$q_0' = \begin{cases} (q_0,1) & \text{if } q_0 \in F \\ (q_0,2) & \text{if } q_0 \notin F \end{cases}$$

and τ' is defined as follows.

(1) If $(q, \varepsilon, z, p, \gamma) \in \tau$, then τ' contains

$$((q,k), \varepsilon, z, (p,l), \gamma) \qquad \text{for } k = 1,\ 2$$

where $l = 1$ if $k = 1$ or $p \in F$, otherwise $l = 2$.

(2) If $(q, a, z, p, \gamma) \in \tau$, where $a \in A$, then τ' contains

$$((q,k), a, z, (p,l), \gamma) \qquad \text{for } k = 2,\ 3$$

where $l = 1$ if $p \in F$, $l = 2$ if $p \notin F$, and τ' also contains

$$((q,1), \varepsilon, z, (q,3), \gamma).$$

Obviously M' is deterministic and has no ε-transitions beginning with a final state. Given a computation of M starting with (q_0, w, z_0), we claim that there is a corresponding computation of M' starting with (q_0', w, z_0), such that if the computation of M ends in state q, then the computation of M' ends in state (q,k), where $k = 1$ or 2. Further, in the computations of M and M', exactly the same word has been read from the tape.

The computation of M' is constructed by induction on the length of the computation of M. Suppose the next step of this computation uses the transition $(q, \varepsilon, z, p, \gamma)$. Then the computation of M' is continued by using the corresponding transition in (1). If the next step in the computation of M uses (q, a, z, p, γ), there are two cases. If M' is in a state $(q,2)$, then the computation of M' is continued by using the transition

$((q,2),a,z,(p,l),\gamma)$ in (2). If M' is in state $(q,1)$, the computation is continued by using $((q,1),\varepsilon,z,(q,3),\gamma)$, followed by $((q,3),a,z,(p,l),\gamma)$.

The computation of M' so constructed will be in a state $(q,1)$ if M has entered a final state since last reading a letter from the tape, and in a state $(q,2)$ otherwise. Note that M' enters a final state when $k=1$ and M reads a letter from the tape.

Suppose $w \in A^*$ and take a letter $a \in A$ (we can assume $A \neq \emptyset$, just by adding a letter to the alphabet of M). Consider the computation of M starting with (q_0,wa,z_0). Since M is deterministic and always scans its entire input, if M accepts w then it must enter a final state after reading the last letter of w and before reading the final a. Then in the corresponding computation of M', M' will enter a final state $(q,3)$ just before reading a, so accepts w. If M does not accept w, then in between reading the last letter of w and reading a, M' remains in states of the form $(q,2)$, so does not enter a final state, hence (being deterministic) does not accept w. Thus $L(M)=L(M')$. □

Note. With minor modification, the argument of the previous lemma can be used to show the complement of a deterministic language is also deterministic. See [20, Theorem 12.1] or [21, Theorem 10.1].

Finally we prove a result on gsm mappings needed in Chapter 5, where the terminology is explained. The proof comes from [20, Theorem 12.3].

Theorem A.5. *The class of deterministic languages is closed under inverse deterministic gsm mappings.*

Proof. Let $S=(Q_S,F_S,A,B,\tau_S,p_0)$ be a deterministic gsm and let L be a deterministic language with alphabet B, so $L=L(M)$ for some deterministic PDA M. If there is a letter in the alphabet of M not in B, we can omit it and any transitions in which it appears. If there is a letter of B, not in the alphabet of M, we can add it to the alphabet. Thus we can assume M has alphabet B, say

$$M=(Q_M,F_M,B,Z,\tau_M,q_0,z_0).$$

By Lemma A.4, we can assume M has no ε-transitions beginning with a final state. Let r be the maximum length of a word $w \in B^*$ such that some edge in the transition diagram of S has label (a,w), for some $a \in A$. We construct a PDA M' recognising $f_S^{-1}(L)$ as follows: $M'=(Q',F',A,Z,\tau',q_0',z_0)$, where:

$$\begin{aligned} Q' &= \{(q,p,w) \mid q \in Q_M,\ p \in Q_S,\ w \in B^* \text{ and } |w| \leq r\} \\ F' &= \{(q,p,\varepsilon) \mid q \in F_M,\ p \in F_S\} \\ q_0' &= (q_0,p_0,\varepsilon) \end{aligned}$$

The transitions in τ' are those specified in (1)–(3) below.

(1) If τ_M contains no transition beginning q,ε,z (where $q \in Q_M$, $z \in Z$), but $(p,a,w,p_1) \in \tau_S$, then τ' contains $((q,p,\varepsilon),a,z,(q,p_1,w),z)$.
(2) If $(q,\varepsilon,z,q_1,\alpha) \in \tau_M$, then τ' contains $((q,p,w),\varepsilon,z,(q_1,p,w),\alpha)$, for all p and w with $(q,p,w) \in Q'$.

(3) If $(q,b,z,q_1,\alpha) \in \tau_M$, where $b \in B$, then τ' contains

$$((q,p,bw),\varepsilon,z,(q_1,p,w),\alpha)$$

for all p and w with $(q,p,bw) \in Q'$.

It is easily seen that M' is deterministic. Suppose M' accepts $u = a_1 \dots a_n$. Then the transitions of type (1) which it uses correspond to transitions of S having the form (p_i,a_i,w_i,p_{i+1}) (because transitions of types (2) and (3) do not change the second coordinate of the state of M'). These transitions give a computation of S with input $a_1 \dots a_n$ and output $w_1 \dots w_n$. The transitions of types (2) and (3) M' uses correspond to transitions of M and give a computation of M in which $w_1 \dots w_n$ is read from its tape, ending, say, in a state q. These transitions do not change the first coordinate of the state of M'. (The third coordinate of the states of M' represents a buffer to receive output from S, using transitions (1); after a letter is read from the buffer, using transitions (3), it is erased.) At the end of the computation M' is in state (p_{n+1},q,ε). Hence $p_{n+1} \in F_S$, so the computation of S is successful, and $a_1 \dots a_n \in f_S^{-1}(w_1 \dots w_n)$. Similarly, $q \in F_M$, so M accepts $w_1 \dots w_n$, that is, $w_1 \dots w_n \in L$. It follows that $L(M') \subseteq f_S^{-1}(L)$.

Conversely, if $a_1 \dots a_n \in f_S^{-1}(L)$, there is a successful computation of S, with input $a_1 \dots a_n$ and output $w = w_1 \dots w_n \in L$, where (a_i,w_i) are the labels on the edges of the corresponding path in the transition diagram. There is a computation of M accepting w. It is left to the reader to construct a computation of M', accepting $a_1 \dots a_n$, from those of S and M. The condition on M, that no ε-transition begins with a final state, is needed because of the possibility that $w_k = \dots w_n = \varepsilon$ for some k. It ensures that, if this happens, M' reads $a_k \dots a_n$ from its tape. Thus $a_1 \dots a_n \in L(M')$, as required. □

Appendix B
The Halting Problem and Universal Turing Machines

Let X be the set of numerical Turing machines, where states are renamed so that the set of states of each machine is $\{2,\dots,r-1\}$ for some r, and where $L=0$, $R=1$. We can define a mapping $gn : X \to \mathbb{N}$ as we did after Theorem 2.19 (but without first modifying the machines). Then gn is a Gödel numbering, that is, it is $1-1$ and its image is recursive (exercise). There is therefore a strictly increasing recursive bijection $f : \mathbb{N} \to gn(X)$. Putting $T_m = gn^{-1}f(m)$, we obtain an enumeration $T_0, T_1, \dots$ of numerical TM's which is effective, in that given m, $f(m) = gn(T_m)$ is computable, and from $gn(T_m)$ one can recover the states, transitions etc. of T_m.

The general halting problem is to give a procedure to decide whether T_m, on input x (i.e. started on tape description $\underline{0}1^x$) halts or not. We shall show this is unsolvable; formally, this means that $B = \{(m,x) \mid T_m \text{ halts on input } x\}$ is not recursive. As in Prop. 3.7, it suffices to show that $A = \{m \mid T_m \text{ halts on input } m\}$ is not recursive.

Suppose A is recursive. Then $\mathbb{N} \setminus A$ is r.e., so $\chi_{p(\mathbb{N}\setminus A)}$ is recursive, hence is computed by a numerical TM T which halts on input m if and only if $\chi_{p(\mathbb{N}\setminus A)}(m)$ is defined, i.e. $m \notin A$, by Cor. 2.22. By renaming, we can assume the set of states of T is $\{2,\dots,r-1\}$ for some r and $L=0$, $R=1$. Then $T = T_p$ for some p.

Then by definition of A, T_p halts on input p if and only if $p \in A$, but T_p halts on input p if and only if $p \notin A$, a contradiction. Hence A is not recursive.

Of course this is related to Prop. 3.7, but is not easy to derive directly from Prop. 3.7 because of the modifications made to the Turing machines and the use of Kleene's Normal Form Theorem.

To make further progress, we shall assume the mapping $g : \mathbb{N} \to \mathbb{N}$ defined by $g(m) = gn(T'_m)$ is recursive (to prove this is a rather complicated exercise). Here, T'_n is the modified TM defined before Lemma 2.18. Taking $n=1$ in Theorem 2.20, the proof shows that $\varphi_{T_m,1}(x) = H(m,x)$, where $H(m,x) = F(g(m),x,\mu t(G(g(m),x,t) = 0))$, where F, G are the functions in Theorem 2.20. Now H is partial recursive, so is computed by a numerical TM, say U, by Cor. 2.22. Then U, started on $Tape(m,x)$ (i.e. $\underline{0}1^m01^x$), gives exactly the same output as T_m on input x. (They either halt with

tape description $\underline{0}1^y$, where $y = \varphi_{T_m,1}(x)$ if this is defined, otherwise they do not halt.) For this reason, U is called a *universal Turing machine*. It is not clear how to construct U from what was done in Chapter 2, but there is a considerable amount of literature on universal Turing machines and their construction, and their relevance to the development of the (stored program) computer. The existence of a universal machine goes back to Turing's original papers ([38], [39].

For a discussion of the halting problem for Turing machines designed to recognise languages, see [20, §7.3].

Appendix C
Cantor's Diagonal Argument

We assume familiarity with the idea of a countable set. Recall that a set is countable if it can be put into one-to-one correspondence with a subset of $\mathbb{N}$. Equivalently, a set C is countable if either $C = \emptyset$ or there is a surjective mapping $f : \mathbb{N} \to C$. In the next theorem, $2^{\mathbb{N}}$ means the set of all subsets of $\mathbb{N}$.

Theorem C.1. *There is no surjective mapping $\mathbb{N} \to 2^{\mathbb{N}}$, consequently $2^{\mathbb{N}}$ is uncountable.*

Proof. Suppose $f : \mathbb{N} \to 2^{\mathbb{N}}$ is surjective; put $Y = \{x \in \mathbb{N} \mid x \notin f(x)\}$. Then $Y = f(z)$ for some $z \in \mathbb{N}$, and $z \in Y \iff z \in f(z) \iff z \notin Y$, by definition of Y, a contradiction. (The argument works for any set X in place of $\mathbb{N}$). □

To interpret this in terms of characteristic functions, we can write

$$Y = \{x \in \mathbb{N} \mid \chi_{f(x)}(x) = 0\}.$$

Then $\chi_Y(x) = 1$ if and only if $\chi_{f(x)}(x) = 0$, that is, $\chi_Y(x) = 1 \dot{-} \chi_{f(x)}(x)$.

Now put $F(m,n) = \chi_{f(m)}(n)$, for $m, n \in \mathbb{N}$. Then for all $x \in \mathbb{N}$,

$$F(z,x) = 1 \dot{-} F(x,x)$$

where $Y = f(z)$, and putting $x = z$ gives a contradiction. The proof is similar to some arguments used in the course (see Props. 3.5 and 3.6) and to the proof of the Gödel Incompleteness Theorem in logic. Writing the values of F as an infinite matrix:

$$\begin{array}{ccccccc}
F(0,0) & F(0,1) & F(0,2) & F(0,3) & F(0,4) & \cdots \\
\searrow & & & & & \\
F(1,0) & F(1,1) & F(1,2) & F(1,3) & \cdots & \\
& \searrow & & & & \\
F(2,0) & F(2,1) & F(2,2) & \cdots & & \\
F(3,0) & F(3,1) & \cdots & & & \\
F(4,0) & \cdots & & & & \\
\vdots & & & & &
\end{array}$$

row m represents the values of $\chi_{f(m)}$, so if f is surjective, each subset of $\mathbb{N}$ is represented by a row; however, $\chi_Y(x) = 1 \dot{-} F(x,x)$, so χ_Y is obtained by changing the values of F on the main diagonal, indicated by the arrows. Then Y is not represented by any row, a contradiction, since a row representing it differs from the first row in the first entry, the second row in the second entry, etc. This explains the name "diagonal argument".

It follows easily that $\mathbb{R}$ is uncountable. Let B be the set of all real numbers a whose decimal expansion has the form $a = 0.a_0a_1\ldots$, where every a_i is either 0 or 1. (In the case of a terminating decimal expansion, add an infinite string of zeros, so $a_1, a_2\ldots$ is always an infinite sequence, which is uniquely determined by a). Every such number $a = 0.a_0a_1\ldots$ defines an element of $2^{\mathbb{N}}$, say X_a, by $\chi_{X_a}(i) = a_i$. The mapping $a \mapsto X_a$ is a bijection from B to $2^{\mathbb{N}}$, hence B is uncountable, and so is $\mathbb{R}$, as a subset of a countable set is countable.

The Russell-Zermelo Paradox

The first proof can be easily adapted to show Cantor's version of set theory is inconsistent; in this set theory, given any predicate P, there is a set $\{x \mid P(x)\}$, such that any object x belongs to the set if and only if $P(x)$ is true. Now let $y = \{x \mid x \notin x\}$. It is easy to see that $y \in y$ if and only if $y \notin y$, a contradiction.

Exercises on Appendix C

1. By explicit use of the diagonal argument, without using $2^{\mathbb{N}}$, show that the subset B of $\mathbb{R}$ is uncountable.
2. Recall from Chapter 2 that Ackermann's function is the function $A : \mathbb{N}^2 \to \mathbb{N}$ defined by

$$\begin{aligned} A(0,y) &= y+1 \\ A(x+1,0) &= A(x,1) \\ A(x+1,y+1) &= A(x,A(x+1,y)) \end{aligned}$$

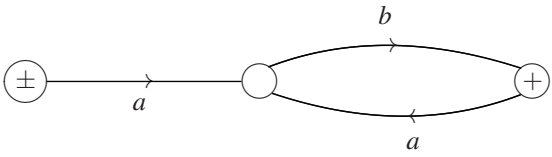
It can be shown that, for any primitive recursive function $f : \mathbb{N}^n \longrightarrow \mathbb{N}$, there exists k with $f(x_1,\ldots,x_n) \leq A(k, \max\{x_1,\ldots,x_n\})$, for all $x_1,\ldots,x_n$. Use this and the diagonal argument to prove that A is not primitive recursive.

Appendix D
Solutions to Selected Exercises

Chapter 1

1. Yes, a derivation is $S, aASb, abSbSb, abSbabb, ababbabb$.

3. A transition diagram for a FSA recognising $\{(ab)^n \mid n = 0,1,2,\ldots\}$ is



5. (ii) No. Otherwise R_L would be of finite index by Theorem 1.7, which implies $1^m01^n0\ R_L\ 1^m01^p0$ for some $n \neq p$, where $m,n,p \geq 0$. Then

$$1^m01^n01^{m+n}\ R_L\ 1^m01^p01^{m+n}$$

a contradiction since $1^m01^n01^{m+n} \in L$, but $1^m01^p01^{m+n} \notin L$.

Chapter 2

1. (a) Let $f_n(x_1,\ldots,x_n) = \max\{x_1,\ldots,x_n\}$, $\underline{x} = (x_1,\ldots,x_n)$. Then

$$\begin{aligned} f_n(\underline{x}) &= \max\{f_{n-1}(x_1,\ldots,x_{n-1}),x_n\} = f_2(f_{n-1}(x_1,\ldots,x_{n-1}),x_n) \\ &= f_2(f_{n-1}(\pi_{1n}(\underline{x}),\ldots,\pi_{n-1,n}(\underline{x})),\pi_{nn}(\underline{x})) \end{aligned}$$

and it suffices by induction on n to show f_2 is primitive recursive. But f_2 has a definition by cases:

$$f_2(x,y) = \begin{cases} \pi_{12}(x,y) & \text{if } x \geq y \\ \pi_{22}(x,y) & \text{if } x < y \end{cases}, \text{ hence } f_2 \text{ is primitive recursive.}$$

4. Clearly $J_1^{-1} = J_1$ is primitive recursive, and J_2^{-1} is primitive recursive by Exercise 3 (b). For $n \geq 2$, if $y = J_{n+1}(x_1,\ldots,x_{n+1})$, then $y = J(x_1, J_n(x_2,\ldots,x_{n+1}))$, so $x_1 = K(y)$ and $J_n(x_2,\ldots,x_{n+1}) = L(y)$, hence $(x_2,\ldots,x_{n+1}) = (J_n^{-1} \circ L)(y)$. Thus $J_{n+1}^{-1} = (K, K_1 \circ L, \ldots, K_n \circ L)$, where $K_1,\ldots,K_n$ are the coordinate functions of J_n^{-1}. It follows by induction on n that J_n^{-1} is primitive recursive for all n. Putting $n = 2$ gives $J_3^{-1} = (K, K \circ L, L \circ L)$.

5. Suppose $\underline{a}_1,\ldots,\underline{a}_k$ are distinct elements of $\mathbb{N}^n$, and $f(\underline{a}_i) = b_i$ (where $b_i \in \mathbb{N}$) for $1 \leq i \leq k$, and $f(\underline{x})$ is undefined for $\underline{x} \notin \{\underline{a}_1,\ldots,\underline{a}_k\}$.

$$\text{Let } g(\underline{x}) = \begin{cases} b_i & \text{if } \underline{x} = \underline{a}_i, \text{ i.e } |\underline{x} - \underline{a}_i| = 0, \text{ for some } i \text{ with } 1 \leq i \leq k \\ 0 & \text{otherwise} \end{cases}$$

Then g is primitive recursive, being obtained from constant functions using a definition by cases. Now let

$$h(\underline{x}) = \mu y(|\underline{x} - \underline{a}_1| \ldots |\underline{x} - \underline{a}_k| = 0)$$

a partial recursive function. Then $h(\underline{a}_i) = 0$ for $1 \leq i \leq k$ and $h(\underline{x})$ is undefined for $\underline{x} \notin \{\underline{a}_1,\ldots,\underline{a}_k\}$. Therefore $f(\underline{x}) = g(\underline{x}) + h(\underline{x})$ is partial recursive.

7. Let H be the iterate of h, so H is primitive recursive by (the easy case of) Question 6. Then $\varphi(x,t,r) = H(x, t \dot{-} r)$, which is obtained from H and known primitive recursive functions by composition.

9. $T_1 = P_1 R^* L P_0$. The effect on the tape description is

$$u01^a0\underline{0}1^c \xrightarrow[P_1]{} u01^a0\underline{1}1^c \xrightarrow[R^*]{} u01^a01^{c+1}\underline{0} \xrightarrow[L]{} u01^a01^c\underline{1} \xrightarrow[P_0]{} u1^a01^c\underline{0}.$$

13. Clearly $T_5 = T_3^{k-1} T_4$ will work.

Chapter 3

1. We assume $A = \{a_1,\ldots,a_n\}$ where $n > 0$ (the case $n = 0$ is easy, as noted in the text). The variables x, y, z are used below to define certain functions, and range over all elements of $\mathbb{N}$. It is a supplementary exercise to verify in detail the claims below that certain functions and predicates are primitive recursive.

(a) Let q be an integer greater than 1. If $r \in \mathbb{N}$, r can be written as

$$r = s_1 + s_2 q + \ldots + s_k q^{k-1}$$

where $0 \leq s_j < q$ for $1 \leq j \leq k$ (by using the division algorithm and induction on r). Putting $s_j = 0$ for $j > k$, the s_j are uniquely determined. To see this, define $Q : \mathbb{N}^3 \to \mathbb{N}$ by primitive recursion:

$$Q(x,y,0) = x$$
$$Q(x,y,z+1) = \text{quo}(y, Q((x,y,z)).$$

and put $F(x,y,z) = \text{rem}(y, Q(x,y,z \dot{-} 1))$, so F is a primitive recursive function. Then the reader can check that $s_j = F(r,q,j)$ for $j \geq 1$. It follows that φ_1 is one-to-one.

Now choosing k as small as possible, k is the least integer m such that such that $r < q^m$, and $k \leq r$ (by induction on r). Define a primitive recursive function $M : \mathbb{N}^2 \to \mathbb{N}$ by $M(x,y) = \mu z \leq x(x < y^z)$, so $k = M(r,q)$. Now put $f(x,z) = F(x,n+1,z)$, $m(x) = M(x,n+1)$. Thus $r = \sum_{j=1}^{m(r)} f(r,j)(n+1)^{j-1}$. From the definition of φ_1

$$r \in \varphi_1(A^*) \Leftrightarrow f(r,j) > 0 \text{ for } 1 \leq j \leq m(r)$$

and the right-hand side is a primitive recursive predicate. Hence φ_1 is a Gödel numbering.

Also, φ_2 is one-to-one by unique factorisation into primes and

$$r \in \varphi_2(A^*) \Leftrightarrow$$
$$(0 < \log_{p_j}(r) \leq n \text{ for } 1 \leq j \leq \log_2(r)) \wedge \left(r = 2^{\log_2(r)} \prod_{j=1}^{\log_2(r)} p_j^{\log_{p_j}(r)} \right).$$

The right-hand side is a primitive recursive predicate, hence φ_2 is a Gödel numbering.

Define $g : \mathbb{N} \to \mathbb{N}$ by $g(r) = 2^{m(r)} \prod_{j=1}^{m(r)} p_j^{f(r,j)}$. Then $g \circ \varphi_1 = \varphi_2$ and g is primitive recursive. If $X \subseteq A^*$ and $\varphi_1(X)$ is r.e. then $\varphi_2(X) = g(\varphi_1(X))$ is r.e. by Lemma 3.3(2).

Now define $g' : \mathbb{N} \to \mathbb{N}$ by $g'(r) = \sum_{j=1}^{\log_2(r)} \log_{p_j}(r)(n+1)^{j-1}$. Then g' is primitive recursive and $g' \circ \varphi_2 = \varphi_1$, so similarly $\varphi_2(X)$ r.e. implies $\varphi_1(X)$ is r.e. Thus $\varphi_1(X)$ is r.e. if and only if $\varphi_2(X)$ is r.e. Applying this to $A^* \setminus X$ and using Lemma 3.8, $\varphi_1(X)$ is recursive if and only if $\varphi_2(X)$ is recursive.

(b) As a hint, suppose $r = s_1 n + \ldots + s_k n^k$ where $1 \leq s_j \leq n$. Then

$$r = n((s_1 - 1) + (s_2 - 1)n + \ldots + (s_k - 1)n^{k-1}) + (n + \ldots + n^k).$$

(d) It is enough to show that $\varphi_2(B^*)$ is recursive, in view of (a) and (b), and in view of (c), we can choose the bijection $\{1, \ldots, n\} \to A$ such that $B = \{a_1, a_2, \ldots, a_s\}$, where $0 \leq s \leq n$. Then

$$r \in \varphi_2(B^*) \Leftrightarrow (r \in \varphi_2(A^*)) \wedge (\log_{p_j}(r) \leq s \text{ for } 1 \leq j \leq \log_2(r))$$
$$\Leftrightarrow (r \in \varphi_2(A^*)) \wedge (\forall j \leq \log_2(r)((j = 0) \vee (\log_{p_j}(r) \leq s))$$

and the right-hand side is a primitive recursive predicate.

2. The construction of some of the TM's is as follows (in all cases, q_0 is the initial state).

R: has set of states $Q = \{q_0, q\}$ and transitions $q_0aqaR \quad (0 \leq a \leq r-1)$.
L: defined similarly, replacing R by L in the transitions.
$\widetilde{R}$: $Q = \{q_0, q, q', h\}$, transitions

$$q_0aq_0aR\ (a \neq 0),\ q_00q0R,\ qaq_0aR\ (a \neq 0),\ q0q'0R,\ q'ahaL$$

where $0 \leq a \leq r-1$.

Chapter 4

1. First, we use Lemma 4.6 to convert the set of productions to

$$S \longrightarrow AA|b$$
$$A \longrightarrow aA|BBB|b$$
$$B \longrightarrow b$$

(The set $\mathscr{U}$ in the proof of Lemma 4.6 is $\{(S,S),(A,A),(B,B),(S,B),(A,B)\}$.) Now, using the procedure in the first part of the proof of Theorem 4.7, we add a new variable C and convert the set of productions to

$$S \longrightarrow AA|b$$
$$A \longrightarrow CA|BBB|b$$
$$B \longrightarrow b$$
$$C \longrightarrow a$$

Then, using the second part of the proof, we add a new variable D and convert the productions to

$$S \longrightarrow AA|b$$
$$A \longrightarrow CA|BD|b$$
$$B \longrightarrow b$$
$$C \longrightarrow a$$
$$D \longrightarrow BB$$

giving the required grammar in Chomsky normal form.

4. (a) Hint: make use of Exercise 3

(c) If you have done parts (a) and (b), you can apply the procedure this gives to the grammar in Example (3), p.3. One possible grammar in the required form, generating $\{0^n1^n \mid n > 0\}$, obtained by this method is

$$G = (\{A, B, S\}, \{0, 1\}, P, S)$$

where P consists of the productions

$$\begin{aligned} S &\longrightarrow 0A|0B \\ A &\longrightarrow S1 \\ B &\longrightarrow 1 \end{aligned}$$

An alternative is to replace P by the set of productions consisting of

$$\begin{aligned} S &\longrightarrow 0A \\ A &\longrightarrow B1|1 \\ B &\longrightarrow 0A \end{aligned}$$

6. A context-free grammar generating $L = \{0^m1^m0^n1^n \mid m,\ n > 0\}$ is

$$G = (\{A, S\}, \{0, 1\}, P, S)$$

where P consists of the productions

$$\begin{aligned} S &\longrightarrow AB \\ A &\longrightarrow 0A1|01 \\ B &\longrightarrow 0B1|01 \end{aligned}$$

To show L is not deterministic, use the Pumping Lemma in the previous exercise.

Chapter 5

1. (a) From $xyx = yxy$, we obtain the consequence $x^2yx^2 = xyxyx$, hence using the other relation, $y^7 = xyxyx = yxy^2x$, so $y^6 = xy^2x$. Since $y^6 = x^4$, we conclude that $x^4 = xy^2x$, hence $x^2 = y^2$, so $y^3 = y^2$ which implies $y = 1$. Now from $xyx = yxy$ it follows that $x^2 = x$, so $x = 1$.

2. The proof, as indicated in the hint, is by induction on n. Let G_n be the group with the given presentation. It is true for $n = 2$ since G_2 is cyclic of order 2, as is S_2. (Indeed, it is true for $n = 1$ as the empty presentation presents the trivial group.) Assume $n > 2$ and $G_{n-1} \cong S_{n-1}$. By Lemma 5.2, there is a homomorphism $G_n \to S_n$ sending x_i to the transposition $(i, i+1)$ for $1 \leq i \leq n-1$,

which is surjective as these transpositions generate S_n (an easy exercise). Hence it suffices to show $|G_n| \leq n!$.
By Lemma 5.2, there is a surjective homomorphism $G_{n-1} \to H$ sending x_i to x_i for $1 \leq i \leq n-2$, hence $|H| \leq (n-1)!$. It is therefore enough to show $(G_n : H) \leq n$. This will follow if we can show that any coset of H is in the set

$$T = \{H, Hx_{n-1}, Hx_{n-1}x_{n-2}, \ldots, Hx_{n-1}x_{n-2}\ldots x_2x_1\}$$

and to do this it suffices to show that if $Hy \in T$, then $Hyx_i^{\pm 1} \in T$ for $1 \leq i \leq n-1$. Since $x_i = x_i^{-1}$, we need to show that $Hx_{n-1}\ldots x_ix_j \in T$ for $1 \leq i \leq n$, $1 \leq j \leq n-1$. (Here $Hx_{n-1}\ldots x_i$ is to be interpreted as H when $i = n$.)
To do this, first note that $x_ix_j = x_jx_i$ if $|i-j| > 1$ and $x_{j-1}x_jx_{j-1} = x_jx_{j-1}x_j$ for $1 < j \leq n-1$.
If $i = j$, then $Hx_{n-1}\ldots x_ix_j = Hx_{n-1}\ldots x_{i+1} \in T$. If $i < j$, then

$$\begin{aligned} Hx_{n-1}\ldots x_ix_j &= Hx_{n-1}\ldots x_jx_{j-1}x_jx_{j-2}\ldots x_i \\ &= H(x_{n-1}\ldots x_{j+1})x_{j-1}(x_jx_{j-1}x_{j-2}\ldots x_i) \\ &= Hx_{j-1}(x_{n-1}\ldots x_i) \\ &= Hx_{n-1}\ldots x_i \in T \end{aligned}$$

since $j-1 \leq n-2$, so $x_{j-1} \in H$.
Finally, if $j < i$, there are two cases. If $j = i-1$, then

$$Hx_{n-1}\ldots x_ix_j = Hx_{n-1}\ldots x_ix_{i-1} \in T.$$

If $j < i-1$, then $Hx_{n-1}\ldots x_ix_j = Hx_jx_{n-1}\ldots x_i = Hx_{n-1}\ldots x_i \in T$ since $j \leq n-2$, so $x_j \in H$.

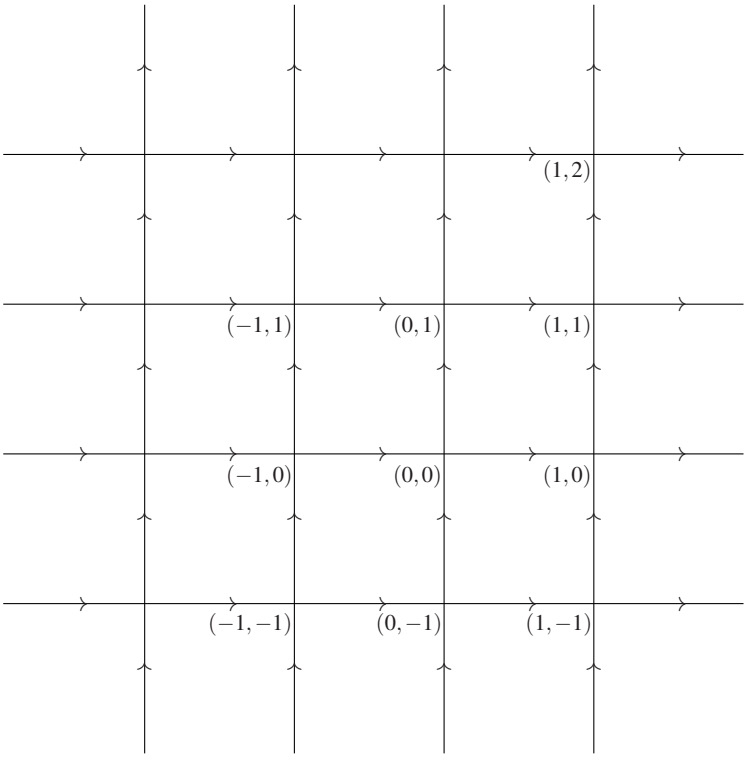
4. Let $f : F(X) \to G$ be the extension of the inclusion mapping $X \to G$ to a homomorphism.
Assume (a) and $\alpha : X \to H$ is a mapping, where G is a group. Then α has a unique extension to a homomorphism $\tilde{\alpha} : F(X) \to H$ by Lemma 5.5. Then αf^{-1} is the unique extension of α to a homomorphism $G \to H$, hence (a) implies (b).
Assume (b). Let $\alpha : X \to F(X)$ be the inclusion map, $\beta : G \to F(X)$ the extension of α to a homomorphism. Then if $g \in G$ is represented by the non-empty reduced word u, $\beta(g) = u \neq 1$ by Lemma 5.4, so $g \neq 1$. Hence (b) implies (c).
Finally the condition on reduced words in (c) implies that $f : F(X) \to G$ has trivial kernel, and if X generates G then f is onto. Hence (c) implies (a).

6. Let X be a basis for F and let Y be a finite set of generators for F. For $y \in Y$, let u_y be a word in $(X^{\pm 1})^*$ representing y. Let X_1 be the finite subset of X consisting of all elements $x \in X$ which occur in u_y (either as x or as x^{-1}) for some $y \in Y$. Then Y is a subset of the subgroup of F generated by X_1 hence F is generated by X_1. By Question 4, no non-empty reduced word in $(X^{\pm 1})^*$ represents the identity element of F, so no non-empty reduced word in $(X_1^{\pm 1})^*$ represents the

identity element. Again by Question 4, X_1 is a finite basis for F. (It follows easily that $X = X_1$.)

8. Suppose F is free with basis X. Let F_x be the subgroup of F generated by x. Then F_x is infinite cyclic by Lemma 5.4, and the inclusion maps $F_x \to F$, for $x \in X$, extend uniquely to a homomorphism $\ast_{x\in X} F_x \to F$. This is an isomorphism by Lemma 5.4 and the normal form theorem for free products. (Alternatively, the inclusion mapping $X \to \ast_{x\in X} F_x$ extends uniquely to a homomorphism $F \to \ast_{x\in X} F_x$; show that this is the inverse map.)
For the converse, show that if F is a free product of infinite cyclic groups, then choosing a generator for each of the infinite cyclic groups gives a basis for F.

9. (b) We can take the free abelian group to be $\mathbb{Z} \times \mathbb{Z}$ with basis $\{x, y\}$, where $x = (1,0)$ and $y = (0,1)$. The Cayley diagram is partly drawn below.



The intersections of the lines represent the vertices (the set of vertices is the set of points in the plane $\mathbb{R}^2$ with integer coordinates). The horizontal arrows have label x and the vertical ones have label y. (Usually one would use

additive notation for this group, but in multiplicative notation, for example, $(1,2) = xy^2 = y^2x$.)
Removing the edges of a finite subgraph always leaves a single infinite component, so the free abelian group of rank 2 has one end.

Appendix C

1. Suppose B is countable. Then there is a surjective mapping $f : \mathbb{N} \to B$. Writing b_{n+1} for $f(n)$, we can write B in a list $B = \{b_1, b_2, \ldots\}$. By definition, we can write

$$\begin{aligned} b_1 &= 0.a_{11}a_{12}a_{13}\ldots \\ b_2 &= 0.a_{21}a_{22}a_{23}\ldots \\ &\vdots \quad \vdots \\ b_i &= 0.a_{i1}\,a_{i2}\,a_{i3}\ldots \\ &\vdots \quad \vdots \end{aligned}$$

where a_{ij} is either 0 or 1, for all integers i, $j \geq 1$. Define $a_i = 1 - a_{ii}$ for $i \geq 1$, then put $b = 0.a_1a_2a_3\ldots$, an element of B since a_i is either 0 or 1 for all i. Therefore $b = b_i$ for some i, which is impossible as the decimal expansions of b and b_i differ in the ith position ($a_i \neq a_{ii}$), a contradiction. Hence B is uncountable.

References

1. J.W. Backus, "The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM conference". In: *Proc. Int. Conf. Inf. Process., Paris 15-20 June 1959,* pp 125–132. UNESCO 1960.
2. M.R. Bridson and R.H. Gilman, "Formal language theory and the geometry of 3-manifolds", *Comment. Math. Helv.***71** (1996), 525–555.
3. D.E. Cohen, *Groups of cohomological dimension one*, Lecture Notes in Mathematics **245**. Berlin, Heidelberg, New York: Springer-Verlag 1972.
4. D.E. Cohen, *Computability and logic.* Chichester: Ellis Horwood; New York etc.: Wiley (Halsted Press) 1987.
5. D.E. Cohen, *Combinatorial group theory: a topological approach*, London Math. Soc. Student Texts **14**. Cambridge: University Press 1989.
6. M.J. Dunwoody, "The accessibility of finitely presented groups", *Invent. Math.* **81** (1985), 449–457.
7. D.B.A. Epstein, J.W. Cannon, D.F. Holt, S.V.F. Levy, M.S. Paterson and W.P. Thurston, *Word processing in groups.* Boston etc.: Jones and Bartlett 1992.
8. S.M. Gersten and H. Short, "Small cancellation theory and automatic groups. II", *Invent. Math.* **105** (1991), 641–662.
9. R.H. Gilman,"Formal languages and infinite groups". In: *Geometric and computational perspectives on infinite groups. Proceedings of a joint DIMACS/Geometry Center workshop, January 3-14, 1994 at the University of Minnesota, Minneapolis, MN, USA and March 17-20, 1994 at DIMACS, Princeton, NJ, USAG* (eds G. Baumslag et al), *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.* **25**, pp. 27–51. Providence, RI: Amer. Math. Soc. 1996.
10. R. Gregorac, "On generalized free products of finite extensions of free groups", *J. London. Math. Soc.* **41** (1966), 662–666.
11. R.H. Haring-Smith, "Groups and simple languages", *Trans. Amer. Math. Soc.***279** (1983), 337–356.
12. M.A. Harrison, *Introduction to formal language theory.* Reading, Mass. etc.: Addison-Wesley 1978.
13. T. Herbst, "On a subclass of context-free groups", *Theor. Inform. Appl.* **25** (1991), 255–272.
14. T. Herbst and R.M. Thomas, "Group presentations, formal languages and characterizations of one-counter groups", *Theoret. Comput. Sci.* **112** (1993), 187–213.
15. G. Higman, B.H. Neumann and H. Neumann, "Embedding theorems for groups", *J. London. Math. Soc.* **24** (1950), 247–254.
16. D.F. Holt and S.E. Rees, "Solving the word problem in real time", *J. London. Math. Soc.* **63** (2001), 623–639.
17. D.F. Holt, S.E. Rees, C.E. Röver and R.M. Thomas, "Groups with context-free co-word problem", *J. London Math. Soc. (2)* **71** (2005), 643–657.

18. D.F. Holt and C.E. Röver, "On real-time word problems", *J. London. Math. Soc.* **67** (2003), 289–301.
19. D.F. Holt and C.E. Röver, "Groups with indexed co-word problem", *Internat. J. Algebra Comput.* **16** (2006), 985–1014.
20. J.E. Hopcroft and J.D. Ullman, *Formal languages and their relation to automata.* Reading, Mass. etc.: Addison-Wesley 1969.
21. J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory, languages and computation.* Reading, Mass. etc.: Addison-Wesley 1979.
22. J.E. Hopcroft, J.D. Ullman and R. Motwani, *Introduction to automata theory, languages, and computation.* Reading, Mass.: Addison-Wesley 2001.
23. A. Karrass and D. Solitar, "The subgroups of a free product of two groups with an amalgamated subgroup", *Trans. Amer. Math. Soc.* **150** (1970), 227–255.
24. A. Karrass, A. Pietrowski and D. Solitar, "Finite and infinite cyclic extensions of free groups", *J. Austral. Math. Soc.* **16** (1973), 458–466.
25. R.C. Lyndon and P.E. Schupp, *Combinatorial group theory*, Ergebnisse der Math. **89**. Berlin, Heidelberg, New York: Springer 1977.
26. W. Magnus, *Noneuclidean tesselations and their groups.* New York, London: Academic Press 1974.
27. D.E. Muller and P.E. Schupp, "Groups, the theory of ends, and context-free languages", *J. Comput. System Sci.* **26** (1983), 295–310.
28. P. Naur et al., "Report on the algorithmic language Algol 60", *Comm. ACM* **3** (1960), 299–314 (ibid. **6** (1963), 1–17).
29. R. Péter, *Recursive functions. 3rd revised ed.* New York, London: Academic Press; Publishing House of the Hungarian Academy of Sciences 1967.
30. D.W. Parkes and R.M. Thomas, "Groups with context-free reduced word problem", *Comm. Algebra* **30** (2002), 3143–3156.
31. J-E. Pin, "Finite semigroups and recognizable languages: An introduction". In: *Semigroups, formal languages and groups, Proceedings of the NATO Advanced Study Institute, York, UK, August 7–21, 1993* (ed. J. Fountain), pp 1–32. Dordrecht: Kluwer Academic Publishers 1995.
32. J.J. Rotman, *An introduction to the theory of groups*, Graduate Texts in Mathematics **148**. New York: Springer-Verlag 1995.
33. G. Rozenberg and A. Salomaa (eds.), *Handbook of formal languages* Vols. 1–3. Berlin: Springer-Verlag 1997.
34. A. Salomaa, *Formal languages* ACM Monograph Series. New York, London: Academic Press [Harcourt Brace Jovanovich] 1973.
35. J.-P. Serre, *Trees.* New York: Springer 1980.
36. M. Shapiro, "A note on context-sensitive languages and word problems", *Internat. J. Algebra Comput.* **4** (1994), 493–497.
37. H. Short, "An introduction to automatic groups". In: *Semigroups, formal languages and groups, Proceedings of the NATO Advanced Study Institute, York, UK, August 7–21, 1993* (ed. J. Fountain), pp 233–253. Dordrecht: Kluwer Academic Publishers 1995.
38. A.M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proc. London Math. Soc. Ser. (2)***42** (1937) 230–265. (Reprinted in *The Undecidable* (Ed. M. David). Hewlett, NY: Raven Press, 1965.)
39. A.M. Turing, "Correction to: On Computable Numbers, with an Application to the Entscheidungsproblem", *Proc. London Math. Soc. Ser. (2)***43** (1938) 544–546.
40. B.L. van der Waerden,"Free products of groups", *Amer. J. Math.* **70** (1948), 527–528.

Index