Inductive case. We suppose that $i > 0$. By induction hypothesis, we have that $A \Rightarrow^* w_1^{i-1}Aw_3^{i-1}$. This together with hypothesis $A \Rightarrow^* w_1 A w_3$ means that

$$\begin{aligned} A &\Rightarrow^* w_1^{i-1}Aw_3^{i-1} \\ &\Rightarrow^* w_1^{i-1}\boldsymbol{w_1 A w_3} w_3^{i-1} \\ &= w_1^i A w_3^i \end{aligned}$$

Also,

$$\begin{aligned} A &\Rightarrow^* w_1^{i-1}Aw_3^{i-1} && \text{(by induction hypothesis)} \\ &\Rightarrow^* w_1^{i-1}w_1 A w_3 w_3^{i-1} && \text{(by hypothesis)} \\ &= w_1^i A w_3^i \\ &\Rightarrow^* w_1^i w_2 w_3^i && \text{(by hypothesis)} \end{aligned}$$

Q.E.D.

We are now ready to present the proof of the Pumping Lemma itself. As in the case of the proof of the Pumping Lemma for Finite-State Automata (Lemma 9.2), we shall use the fact that if language L is infinite, then there is no upper bound on the length of words in L.

LEMMA 10.3 (A Pumping Lemma for Context-Free Languages): Let L be an infinite context-free language. Then L must contain a word $z = uvwxy$ such that:

(i) v and x are not both ε.

(ii) uv^iwx^iy is in L for all $i \geq 0$.

PROOF If L is an infinite context-free language, then, by Corollary 10.2, so is $L\backslash\{\varepsilon\}$. Furthermore, by Theorem 10.3, there exists a Chomsky normal form grammar G generating $L\backslash\{\varepsilon\}$. Suppose grammar G has exactly k nonterminals and set $m = 2^{k-1}$. Since L is infinite, L must contain a word z whose length exceeds m. That is, $|z| > 2^{k-1}$, and proposition (1) above implies that any parse tree $\mathcal{T}$ for z must contain a path of length at least $k+1$. Such a path will have at least $k+2$ nodes. Since all $k+1$ interior nodes in such a path are labeled by nonterminals, the Pigeonhole Principle (Theorem 0.6) tells us that at least one nonterminal must be repeated along such a path.

Suppose that P is a path of maximal length in $\mathcal{T}$. (There may be more than one path of this length in $\mathcal{T}$.) By the foregoing, there must be at least $k+2$ nodes in P and at least one nonterminal A, say, must occur more than once—at nodes n_1 and n_2, say (see Figure 10.7.4). We assume that n_1 is higher on path P than n_2. We can also assume without loss of generality that the subpath P_1 of P from n_1 to leaf is of length $\leq k+1$.[1]

[1]To see why this is so, consider the following. Imagine traversing P from below starting with its leaf and keeping track of encountered labels. Of the first $k+2$ nodes encountered, only the leaf has a terminal label. Consequently, the next $k+1$ nodes as we proceed upward along P must include repeated nonterminal labels. So if the given subpath P_1 is not of length $\leq k+1$, it can be replaced by a subpath that *is* of length $\leq k+1$.

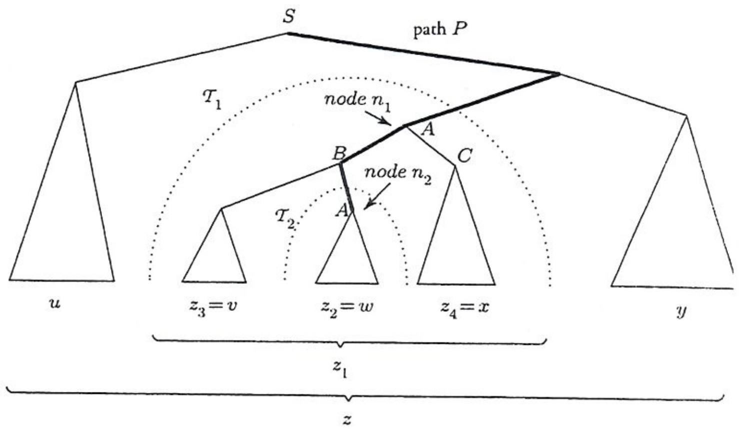


Figure 10.7.4 A Parse Tree $\mathcal{T}$ for Word z with $|z| > 2^{k-1}$, where Context-Free Grammar G has k Nonterminals.

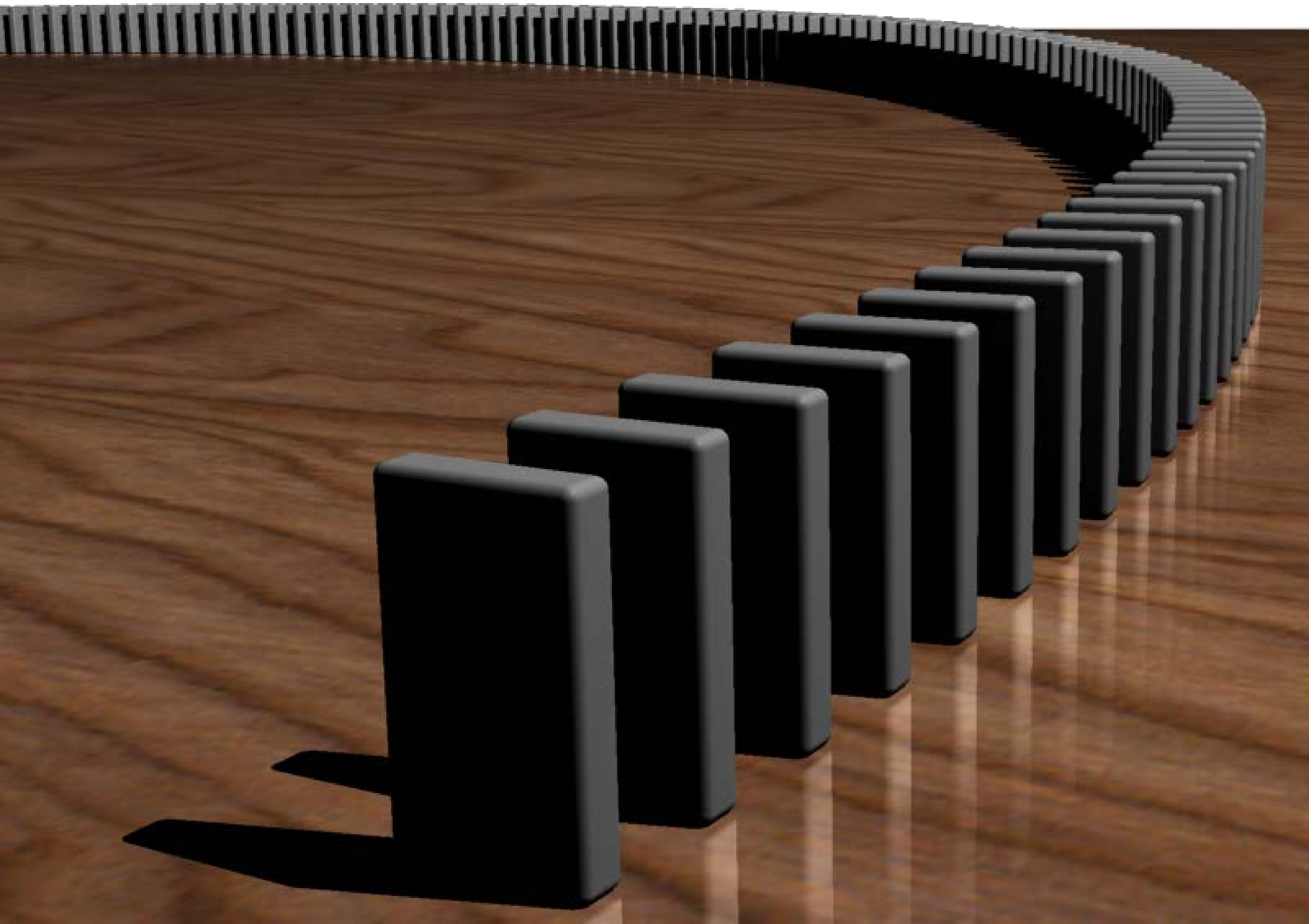
- Let $\mathcal{T}_1$ be the subtree of $\mathcal{T}$ with root n_1. Let z_1 be the word labeling the leaves of $\mathcal{T}_1$. Since P is a path of maximal length in $\mathcal{T}$, it follows that P_1 is a path of maximal length in $\mathcal{T}_1$. But P_1 has length $\leq k+1$. So by proposition (3), we see that $|z_1|$ is $\leq 2^k$.
- Similarly, let $\mathcal{T}_2$ be the subtree whose root is n_2, and suppose that word z_2, say, labels its leaves. Then we have $z_1 = z_3z_2z_4$, where z_3 and z_4 cannot both be ε. Why not? Well, the production used to expand node n_1 must be of the form $A \rightarrow BC$ for some nonterminals B and C. But then the subtree $\mathcal{T}_2$ must be entirely contained in either the subtree whose root is labeled by B or the subtree whose root is labeled by C. Let us suppose that $\mathcal{T}_2$ is within the B subtree, as in Figure 10.7.4. Then, by the fact that G is in Chomsky normal form, nonterminal C cannot generate ε, which means that z_4 is nonempty. Similarly, if $\mathcal{T}_2$ is entirely contained in the C subtree, then z_3 must be nonempty. Consequently, we have $A \Rightarrow^* z_3Az_4$ with z_3 and z_4 not both ε.
- Also, since A is the root of subtree $\mathcal{T}_2$, we have $A \Rightarrow^* z_2$. But now it follows, by Lemma 10.2, that $A \Rightarrow^* z_3^iz_2z_4^i$ for all $i \geq 0$. Letting $v = z_3$, $w = z_2$, and $x = z_4$, we see that z is of the form $uvwxy$, which completes the proof. (For the identity of words u and y, see Figure 10.7.4.)

Q.E.D.

Just as the purpose of the Pumping Lemma for Finite-State Automata was to demonstrate the existence of nonregular languages, so we shall now use our Pumping Lemma for Context-Free Languages to demonstrate the existence of languages that are not context-free. As a first example, we shall show that $L = \{a^nb^nc^n | n > 0\}$ is not context-free.

THEOREM 10.10: The language $L = \{a^nb^nc^n | n > 0\}$ is not context-free. Thus, there exists a Turing-acceptable language that is not context-free.

PROOF (indirect). We begin by supposing, for the sake of producing a contradiction, that L is context-free. Since L is evidently infinite, the Pumping Lemma applies: There must exist a word $uvwxy$ such that v and x are not both empty and v and x can be pumped. The cases whereby v or x or both



instruction sequence proceeds in accordance with the definition or state diagram of M. Finally, M halts reading a single *1* on an otherwise blank tape. Now, in the context of Chapter 1, M's halting meant M's being in some state q_t reading some symbol s_j such that M has no instruction for the case where it is in state q_t scanning symbol s_j. This is still precisely what we shall assume here. However, it will be convenient to introduce an alternative *description* of what it is for M to halt, which we now elaborate.

First, let us rule out so-called *useless instructions*—that is, instructions whose execution causes no changes in the current state of the machine, the location of the read/write head, or the contents of the tape. An example of such an instruction would be the quadruple $q_0, a; a, q_0$. This amounts to a slight change in the definition of Turing machines as given in Definition 1.1. However, it is a change that has no significant theoretical consequences.

We shall further assume that instructions are instantaneously executed at a uniform rate of one per time unit as marked by some clock. Thus, if, at time $t = 0$, M is in state q_0 scanning the leftmost symbol of $w = aabb$ on an otherwise blank tape

$$\ldots Bq_0aabbB\ldots$$

then, after executing the instruction $\langle q_0, a; \text{R}, q_1\rangle$, the state of M and the tape contents at time $t = 1$ will be completely described by the machine configuration description

$$\ldots Baq_1abbB\ldots$$

Computations may be of finite or infinite length, where the length of a computation is just the length of the corresponding instruction sequence. Evidently, any accepting computation is of finite length. But we shall permit ourselves to speak of M's machine configuration description at time $t = n+1, n+2, \ldots$ even if M's computation was of length n—that is, even if M halted at time $t = n$. Since we are ruling out useless instructions, it follows that M halts if and only if, for some n, its configuration description for time $t = n$ is identical with its configuration description at time $t = n+1$. So M halts at or before time $t = n$ provided that M's configuration at time $t = n$ is identical to its configuration at time $t = n+1$.

Having completed the necessary preliminaries, we can now proceed to give the actual proof of

THEOREM 8.9 (Cook–Levin Theorem): The Satisfiability Problem for CNFs is NP-hard and hence NP-complete.

PROOF By the definition of NP-hardness, we need to show that, for an arbitrary language $L \in NP$, we have that L is polynomial-time reducible to $L_{\mathcal{CNFS}at}$—that is, $L \leq_p L_{\mathcal{CNFS}at}$. So we start by letting L be an arbitrary member of NP. By the definition of NP, we can assume that there exists a nondeterministic, single-tape Turing machine M that accepts L and a polynomial $p(n)$ such that, if $w \in L$, then there exists an accepting computation of M for input w whose length is $O(p(|w|))$. For such an M, p, and w, we shall describe the construction of a sentence $\mathcal{F}_w$ of propositional logic that is a CNF and that models or characterizes the class $\mathcal{C}_w$ of accepting computations of M for input w. Intuitively, $\mathcal{F}_w$ will say that:

- M starts scanning the leftmost symbol of w on a tape that contains w and is otherwise blank.
- All executed transitions are in accordance with M's transition diagram.
- M halts, after no more than $O(p(|w|))$ steps, scanning a *1* on an otherwise blank tape.

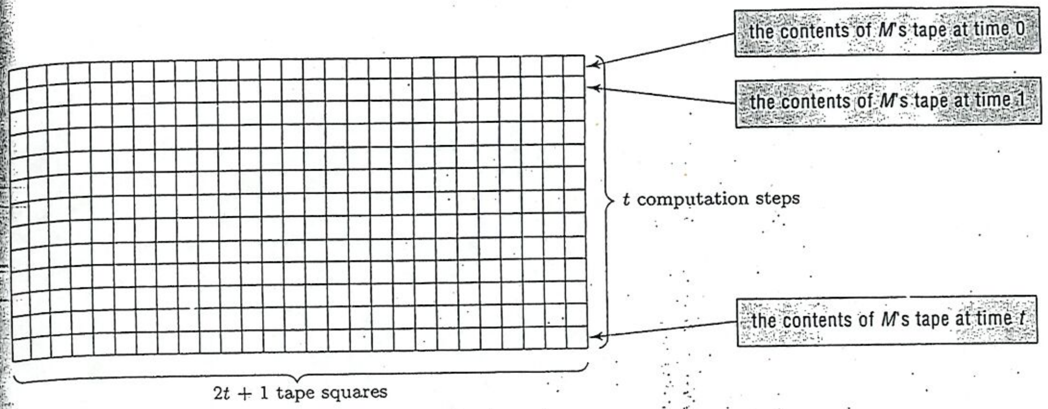

the contents of M's tape at time 0
the contents of M's tape at time 1
t computation steps
the contents of M's tape at time t
$2t + 1$ tape squares

Figure 8.6.1

In other words, by the nature of the construction of $\mathcal{F}_w$, it will be true that

That is, $\mathcal{F}_w$ is satisfiable. $\longleftarrow$ **$\mathcal{F}_w$ is in $L_{\mathcal{CNFS}at}$ $\Leftrightarrow$ $\mathcal{C}_w$ is nonempty $\Leftrightarrow$ M accepts w** $\longrightarrow$ That is, there exists an accepting computation of M for input w.

In addition, we shall show that the construction of $\mathcal{F}_w$ from M, p, and w can be carried out by a deterministic Turing machine M', the number of steps in whose computation for input w is $O(p'(|w|))$ for some polynomial p'. As usual, our argument for the existence of M' will be informal: We shall appeal to the reader's general sense of the capacities and resource needs of Turing machines.

In what follows, let us assume a given input word w. Furthermore, let $t =_{\text{}} p(|w|)$ and assume that $t = p(|w|) \geq |w|$, where p is the given polynomial by virtue of which L is in NP.[2] If w is accepted by M, then, by hypothesis, this occurs in $O(p(|w|))$ steps. (Expressed differently, we are assuming that $time_M(|w|)$ is $O(p(|w|))$.) Without loss of generality, we may assume that acceptance occurs in $t = p(|w|)$ or fewer steps (see Exercise 0.5.13). So to determine whether w is accepted, we need only start M running on input w and observe whether, after t or fewer steps, M has halted in an accepting configuration. Moreover, since M's tape head can at most move one square to the right or one square to the left during any one step, it can move, over the course of its entire computation, at most t steps to the right or t steps to the left of its initial square. This means that at most $t + 1 + t = 2t + 1$ squares of the tape can be scanned over the course of M's entire computation for input w. (In other words, $space_M(|w|) \leq 2t + 1$.) The squares on this portion of M's tape are assigned positions ranging from 1 to $2t + 1$, inclusive. In any case, M's successive tape configurations over its entire computation for input w may be described as a two-dimensional array with $t + 1$ rows and $2t + 1$ columns (see Figure 8.6.1). For example, the first line of the array of Figure 8.6.1 will contain the string $B^t wB^{t-|w|+1}$, corresponding to M's initial tape contents. We shall assume that M has $m+1$ states and that tape alphabet $\Gamma = \{\alpha_1, \ldots, \alpha_r\}$ with input alphabet $\Sigma \subseteq \Gamma$. It will prove convenient to write α_0 for the blank B and α_1 for symbol *1*.

[2] If the given polynomial p does not satisfy this condition, then replace p with p_1 defined as

$$p_1(n) = p(n) + n$$

which is a polynomial in n with $p_1(n) \geq n$ for all n.