the length of the list. Since the list of accessible ID's has length no greater than $d^{f(n)}$ times the length of an ID, which can be encoded in $1 + k(f(n) + 1)$ symbols, the time is bounded by $c^{f(n)}$ for some constant c. □

Theorem 12.11 (*Savitch's theorem*) If L is in NSPACE($S(n)$), then L is in DSPACE($S^2(n)$) provided $S(n)$ is fully space constructible and $S(n) \geq \log_2 n$.

Proof Let $L = L(M_1)$, where M_1 is an $S(n)$ space-bounded nondeterministic TM. For some constant c, there are at most $c^{S(n)}$ ID's for an input of length n. Thus, if M_1 accepts its input, it does so by some sequence of at most $c^{S(n)}$ moves, since no ID is repeated in the shortest computation of M_1 leading to acceptance.

Let $I_1 \vdash^{(i)} I_2$ denote that the ID I_2 can be reached from I_1 by a sequence of at most 2^i moves. For $i \geq 1$, we can determine if $I_1 \vdash^{(i)} I_2$ by testing each I' to see if $I_1 \vdash^{(i-1)} I'$ and $I' \vdash^{(i-1)} I_2$. Thus the space needed to determine if we can get from one ID to another in 2^i moves is equal to the space needed to record the ID I' currently being tested plus the space needed to determine if we can get from one ID to another in 2^{i-1} moves. Observe that the space used to test whether one ID is reachable from another in 2^{i-1} moves can be reused for each such test.

The details for testing if w is in $L(M_1)$ are given in Fig. 12.5. The algorithm of Fig. 12.5 may be implemented on a Turing machine M_2 that uses a tape as a stack of activation records† for the calls to TEST. Each call has an activation record in which the values of parameters I_1, I_2, and i are placed, as well as the value of local variable I'. As I_1, I_2 and I' are ID's with no more than $S(n)$ cells, we can represent each of them in $S(n)$ space. The input head position in binary uses $\log n \leq S(n)$ cells. Note that the input tape in all ID's is fixed and is the same as the input to

```
begin
    let |w| = n and m = ⌈log2 c⌉;
    let I0 be the initial ID of M1 with input w;
    for each final ID If of length at most S(n) do
        if TEST (I0, If, mS(n)) then accept;
end;

procedure TEST (I1, I2, i);
    if i = 0 and (I1 = I2 or I1 ⊢ I2) then return true;
    if i ≥ 1 then
        for each ID I' of length at most S(n) do
            if TEST (I1, I', i − 1) and TEST (I', I2, i − 1) then
                return true;
    return false
end TEST
```

Fig. 12.5 Algorithm to simulate M_1.

† An "activation record" is the area used for the data belonging to one call of one procedure.

M_2, so we need not copy the input in each ID. The parameter i can be coded in binary using at most $mS(n)$ cells. Thus each activation record takes space $0(S(n))$.

As the third parameter decreases by one each time TEST is called, the initial call has $i = mS(n)$, and no call is made when i reaches zero, the maximum number of activation records on the stack is $0(S(n))$. Thus the total space used is $0(S^2(n))$, and by Theorem 12.1, we may redesign M_2 to make the space be exactly $S^2(n)$.

Example 12.4

$$\text{NSPACE}(\log n) \subseteq \text{DSPACE}(\log^2 n)$$

$$\text{NSPACE}(n^2) \subseteq \text{DSPACE}(n^4) \quad \text{and} \quad \text{NSPACE}(2^n) \subseteq \text{DSPACE}(4^n).$$

Note that for $S(n) \geq n$, Savitch's theorem holds even if $S(n)$ is space constructible rather than fully space constructible. M_2 begins by simulating a TM M that constructs $S(n)$, on each input of length n, taking the largest amount of space used as $S(n)$ and using this length to lay out the space for the activation records. Observe, however, that if we have no way of computing $S(n)$ in even $S^2(n)$ space, then we cannot cycle through all possible values of I_f or I' without getting some that take too much space.

12.5 TRANSLATIONAL LEMMAS AND NONDETERMINISTIC HIERARCHIES

In Theorems 12.8 and 12.9 we saw that the deterministic space and time hierarchies were very dense. It would appear that corresponding hierarchies for nondeterministic machines would require an increase of a square for space and an exponential for time, to simulate a nondeterministic machine for diagonalization purposes. However, a translational argument can be used to give a much denser hierarchy for nondeterministic machines. We illustrate the technique for space.

A translation l.....

The first step is to show that containment translates upward. For example, suppose it happened to be true (which it is not) that $\text{NSPACE}(n^3) \subseteq \text{NSPACE}(n^2)$. This relation could be translated upward by replacing n by n^2, yielding

$$\text{NSPACE}(n^6) \subseteq \text{NSPACE}(n^4).$$

Lemma 12.2 Let $S_1(n)$, $S_2(n)$, and $f(n)$ be fully space constructible, with $S_2(n) \geq n$ and $f(n) \geq n$. Then

$$\text{NSPACE}(S_1(n)) \subseteq \text{NSPACE}(S_2(n))$$

nplies

$$\text{NSPACE}(S_1(f(n))) \subseteq \text{NSPACE}(S_2(f(n))).$$