Let M be a nondeterministic machine that always halts and runs in space $S(n)$. Then, M runs in at most $T = qng^{S(n)}$ steps for q states and g symbols in the tape alphabet. Now, if necessary, reprogram M to erase the work tape and end in a particular state when it accepts.

As before, a configuration records the contents of the work tape, the state of the machine, and the position of the head on the input tape. To simplify things we pad each configuration to have length $S(n)$. So, there is a unique starting configuration C_s for a given input and a unique accepting configuration C_a. The question is: Is there a legal sequence of moves from C_s to C_a?

Define the boolean function f that takes as arguments two configurations C_1 and C_2 and an integer i such that $f(C_1, C_2, i)$ is TRUE if M can get from C_1 to C_2 in at most i steps and FALSE otherwise. The key is that this function can be computed recursively as follows:

```
Calculating $f(C_1, C_2, i)$:

if $i = 1$ then return $oneStep(M, C_1, C_2)$
else {
  for all strings $C_3$ of length $S(n)$ {
   if $f(C_1, C_3, i/2)$ = TRUE and $f(C_3, C_2, i/2)$ = TRUE
    then return TRUE
  }
  failing which : return FALSE
}
```

The boolean function *oneStep* determines whether configuration C_1 follows from configuration C_2 according to the rules of M.

So, we build a deterministic machine and ask it for the answer to $f(C_s, C_a, T)$. How much storage does this use? The recursive program can use a stack for storage, pushing a "snapshot" of the subroutine's variables each time it calls itself again. Each configuration and hence each snapshot uses $O(S(n))$ space. The maximum number of snapshots on the stack at any one time is $\log T$: at each stage of the recursion, the integer is halved so that the recursion goes down only $\log T$ levels. Because $\log T$ is $O(S(n))$, it follows that the total space needed is $O((S(n))^2)$ cells.

The statement of the theorem now follows.