In this lecture, we are going to prove the following result:

\[ N_{space}(s(n)) = Co - N_{space}(s(n)) \]

Let \( A = L(M) \), where \( M \) is a nondeterministic Turing machine, using space \( s(n) \).

Define \( C_k(x) \) to be the number of configurations of \( M \) that are reachable from the start configuration in steps less than \( k \) on input \( x \). Before we proceed our proof, we need two assumptions about \( M \): (1) \( M \) counts the numbers of steps it has run; (2) \( M \) runs for less than \( 2^{b \times (s(n))} \) steps (Notice that \( C_k(x) \) can be written in binary in \( O(s(n)) \) space).

The key point is to show that \( C_k(x) \) is computable in \( N_{space}(s(n) + \log(k)) \).

Assume that \( C_k(x) \) is computable in \( N_{space}(s(n)) \). The following little code will show that \( \text{Nondeterministic space is closed under complementation.} \)

**Begin**

On input \( x \)
- Compute \( C_{2^{b \times (s(n))}}(x) \);
- \( Count := 0; \)
- \( Flag := \text{False}; \)
- For each configuration \( c \) using \( s(n) \) space, Guess a path from the initial configuration to \( c \). If a path is found, then
  - \( Count := Count + 1; \)
  - If \( c \) is accepting, then
    - \( Flag := \text{True}; \)
- EndFor
- If \( Count \neq C_{2^{b \times (s(n))}}(x) \), then reject; else accept iff \( Flag = \text{False} \)

**End**

The code says that the exact number of configurations of size \( s(n) \) reachable by \( M \) from \( \text{START} \) can be computed, then we can test in \( N_{space}(s(n)) \) if \( M \) rejects.

Now the remaining thing is to prove that \( C_k(x) \) is computable. We also write a little code for it (the basic idea is inductive counting).

**Begin**

On input \( x \)
- \( C_0(x) := 1; \)
- Compute \( C_{k+1}(x) \) from \( C_k(x) \):
  - \( c_{k+1} := 1; \)
  - For each configuration \( c \)
    - For each configuration \( d \) such that there is a path between them, if \( d \) is reachable in less than \( k \) steps, then
      - \( Count := 0; \)
      - \( Flag := \text{False}; \)
      - For each configuration \( e \), guess a path from the initial configuration to \( e \) of less than \( k \). If a path is found, then
        - \( Count := Count + 1; \)
        - If \( e = d \), then
          - \( Flag := \text{True}; \)
        EndFor
      - If \( Count \neq C_k(x) \), then halts and rejects else if \( Flag := \text{True} \), then
        - \( C_{k+1} := C_{k+1} + 1 \)
        Go to next \( c \)
    EndFor
- If \( Count \neq C_k(x) \), then halts and rejects else if \( Flag := \text{True} \), then
  - \( C_{k+1} := C_{k+1} + 1 \)
  Go to next \( c \)

**End**

The ideas used in the above code is the mathematical induction. It shows how to find \( C_{k+1}(x) \) from \( C_k(x) \).