TIME(f(n)) the class (set) of languages that can be decided by a single tape, deterministic Turing machine in time O(f(n)) where n is the number of cells used by the input to the Turing machine.

NTIME(f(n)) the same but non-deterministic Turing machines

P = union over all k of TIME(nk)

NP = union over all k of NTIME(nk)

EXPTIME = union over all k of TIME(2^(nk))

P is a subset of NP

P is a subset of EXPTIME

SPACE(f(n)) the class of languages that can be decided by a single tape, deterministic Turing machine that uses at most O(f(n)) cells in its execution.

NSPACE(f(n)) the same but for a nondeterministic Turing machines

PSPACE = union over all k of SPACE(nk)

NPSPACE = union over all k of NSPACE(nk)

1) Prove that P is a subset of PSPACE.

Since P runs in time nk, it can’t hit more than nk cells in that time, so it uses at most nk cells.

2) NP is a subset of NPSPACE

3) Prove that PSPACE is a subset of EXPTIME.

Consider a configuration of the PSPACE Turing machine. (what state is it in, what cell is it looking at, and what is the contents of the tape)

Each configuration is size O(nk). Because the machine uses at most O(nk) cells.

How many possible configurations are there? Let d = # of states, let m = # of tape symbols. m^(O(nk)) \* d \*O(nk) and this is 2^(O(zk) for some z and k.

There are only an exponential number of configurations. Suppose the machine is not in EXPTIME. It runs time longer than 2^(O(zk)). By the Pigeonhole Principle, we have repeated a configuration so the machine is looping. If the machine is in an infinite loop, this contradicts the claim that the machine decides the language.

P subset of PSPACE subset of EXPTIME

NP subset of NPSPACE subset of NEXPTIME

Savich’s Theorem: NSPACE(f(n)) is a subset of SPACE(f2(n))

(A nondeterministic machine that uses O(f(n)) cells can be simulated by a deterministic machine that uses O(f2(n)) cells.)

PSPACE = NPSPACE

Proof: Consider all the configurations of a nondeterministic Turing machine that runs using at most O(f(n)) cells. There are an exponential number of such configurations.

We need to have a deterministic Turing machine that asks if we can go from the initial configuration to some configuration with qaccept ?

(Let’s assume there is only one configuration with qaccept.).

The deterministic machine will encode the function

CANYIELD(c1, c2, t) = true if the non-deterministic machine can go from configuration c1 to configuration c2 in t moves.

CANYIELD(c1, c2, t) =

If c1 = c2 then return true

For each configuration cx,

CANYIELD(c1, cx t/2)

CANYIELD(cx, c2, t/2)

until both return true

If both return true for some cx, return true

If all cx are tried without both returning true, return false.

Each configuration is size O(f(n)). The running time of the nondeterministic machine could be 2O(f(n)).

CANYIELD((initial config.), (accept config), t=2O(f(n)))

The height of my call stack is log2(2O(f(n))) = O(f(n))

The total space we need is O(f(n) x f(n)) = O(f2(n))

PSPACE = NPSPACE. This means NP is a subset of PSPACE and NPSPACE is a subset of EXPTIME.