$PSPACE$ Completeness

Definition: A language Q is $PSPACE$**-complete** if it satisfies two conditions:

1. Q is in $PSPACE$,
2. every L in $PSPACE$ is polynomial time reducible to Q.

If Q merely satisfies condition 2 we say that it is $PSPACE$**-hard**.

NOTE: We chose polynomial time reduction here because we want the reductions to be "easy".

$PSPACE$ Completeness

The canonical example of a $PSPACE$-complete problem is a generalization of SAT using formulas with quantifiers,

$$\forall x \exists y[(x \vee y) \wedge (\overline{x} \wedge \overline{y})]$$

Then the language

$$TQBF = \{\langle \phi \rangle | \phi \text{ is a true fully quantified Boolean formula}\}.$$

$PSPACE$ Completeness

Theorem:

$TQBF$ is $PSPACE$-complete.

Proof: First we show that it is in $PSPACE$.

$T =$ "On input $\langle \phi \rangle$, a fully quantified Boolean formula:

1. If ϕ contains no quantifiers, then it is an expression with only constants, so evaluate ϕ and *accept* if it is true; otherwise, *reject*.
2. If ϕ equals $\exists x\ \psi$, recursively call T on ψ, first with 0 substituted for x and then with 1 substituted for x. If either result is accept, then *accept*; otherwise, *reject*.
3. If ϕ equals $\forall x\ \psi$, recursively call T on ψ, first with 0 substituted for x and then with 1 substituted for x. If both results are accept, then *accept*; otherwise, *reject*."

Observe that the machine runs in linear space.

For the second proof obligation it is possible to show that each language $L \in PSPACE$ can by polynomial time reduced to $TQBF$ by encoding every string in L as a quantified formula. This is similar to the completeness proof of the SAT problem.

L and NL

Definition: L is the class of languages that are decidable in logarithmic space on a deterministic Turing machine,

$$L = SPACE(\log n).$$

NL is the class of languages that are decidable in logarithmic space on a nondeterministic Turing machine,

$$NL = NSPACE(\log n).$$

Context-free Languages

Theorem: Let $A = \{0^k 1^k | k \geq 0\}$, then

$$A \in L.$$

Proof: Observe that a binary counter can store a value $v \leq 2^p$ in $\log 2^p = p$ bits. This allows us to build a machine that recognizes this language with two binary counters such that $k \leq 2^n$, thus using $O(\log n)$ space.

$PATH$

Theorem: Let

$$PATH = \{\langle G, s, t\rangle | G \text{ is a directed graph with a path from } s \text{ to } t\}$$

then

$$PATH \in NL$$

Proof: The machine only stores a pointer to the current node and runs a maximum of m iterations where m is the number of nodes. At each iteration it nondeterministically chooses which node to visit next. This machine runs in $\log n$ space where $n = 2^p$ and p is the number of bits required to count up to $m \leq n$.

Log Space Transducers

Definition:

A ***log space transducer*** is a Turing machine with a read-only input tape, a write-only output tape, and a read/write work tape. The work tape may contain $O(\log n)$ symbols. A log space transducer M computes a function $f\colon \Sigma^* \longrightarrow \Sigma^*$, where $f(w)$ is the string remaining on the output tape after M halts when it is started with w on its input tape. We call f a ***log space computable function***. Language A is ***log space reducible*** to language B, written $A \leq_{\mathrm{L}} B$, if A is mapping reducible to B by means of a log space computable function f.

Definition: A language Q is NL-complete if

1. $Q \in NL$,
2. every $A \in NL$ is log space reducible to Q.

$PATH$

Theorem:

$$PATH \in NL - \text{complete.}$$

(see book for proof.)

Recall that $PATH \in P$, from this and the above it follows that

$$NL \subseteq P$$

(all languages in NL can be reduced to $PATH$ which in turn is in P)

Putting it all together

Putting this all together gives us the following hierarchy:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$