Part II
Mysteries

We now enter Terra Incognita by extending deterministic computations with tools like random choices, nondeterministic guesses, etc., the power of which is completely unknown. Yet many fascinating discoveries were made there in which we will proceed to take a glimpse.

4 Nondeterminism; Inverting Functions; Reductions

4.1 An Example of a Narrow Computation: Inverting a Function

Consider a P-time function F. For convenience, assume $\|F(x)\| = \|x\|$, (often $\|F(x)\| = \|x\|^{\Theta(1)}$ suffices). Inverting F means finding, for a given y, at least one $x \in F^{-1}(y)$, i.e. such that $F(x) = y$.

We may try all possible x for $F(x) = y$. Assume F runs in linear time on a Pointer Machine. What is the cost of inverting F? The *space* used is $\|x\| + \|y\| + \text{space}_F(x) = O(\|x\|)$. But time is $O(\|x\| 2^{\|x\|})$: absolutely infeasible. No method is currently proven much better in the worst case. And neither could we prove some inversion problems to require *super-linear* time. This is the sad present state of Computer Science!

An Example: Factoring. Let $F(x_1, x_2) = x_1 x_2$ be the product of integers. For simplicity, assume x_1, x_2 are primes. A fast algorithm in sec. 5.1 determines if an integer is prime. If not, no factor is given, only its existence. To invert F means to factor $F(x)$. The density of n-bit primes is $\approx 1/(n \ln 2)$. So, factoring by exhaustive search takes *exponential* time! In fact, even the best known algorithms for this ancient problem run in time about $2^{\sqrt{\|y\|}}$, despite centuries of efforts by most brilliant people. The task is now commonly believed infeasible and the security of many famous cryptographic schemes depends on this *unproven* faith.

One-Way Functions: $F : x \to y$ are those easy to compute ($x \mapsto y$) and hard to invert ($y \mapsto x$) for most x. Even their existence is sort of a religious belief in Computer Theory. It is unproven, though many functions *seem* to be one-way. Some functions, however, are proven to be one-way, IFF one-way functions EXIST. Many theories and applications are based on this hypothetical existence.

Search and NP Problems

Let us compare the inversion problems with another type – the search problems specified by computable in time $\|x\|^{O(1)}$ relations $F(x, w)$: given x, find w s.t. $F(x, w)$. There are two parts to a search problem: (a) decision problem: decide if w (called ***witness***) exist, and (b) a constructive problem: actually find w.

Any inversion problem is a search problem and any search problem can be restated as an inversion problem. E.g., finding a Hamiltonian cycle C in a graph G, can be stated as inverting a $f(G, C)$, which outputs $G, 0 \ldots 0$ if C is in fact a Hamiltonian cycle of G. Otherwise, $f(G, C) = 0 \ldots 0$.

Similarly any search problem can be reduced to another one equivalent to its decision version. For instance, factoring x reduces to bounded factoring: given x, b find p, q such that $pq = x$, $p \le b$ (where decisions yield construction by binary search).

Exercise: Generalize the two above examples to reduce any search problem to an inverting problem and to a decision problem.

The ***language*** of a problem is the set of all acceptable inputs. For an inversion problem it is the range of f. For a search problem it is the set of all x s.t. $F(x, w)$ holds for some w. An ***NP language*** is the set of all inputs acceptable by a P-time ***non-deterministic*** Turing Machine (sec. 3.3). All three classes of languages – search, inversion and NP – coincide (NP $\iff$ search is straightforward).

Interestingly, polynomial *space* bounded deterministic and non-deterministic TMs have equivalent power. It is easy to modify TM to have a unique accepting configuration. Any acceptable string will be accepted in time 2^s, where s is the space bound. Then we need to check $A(x, w, s, k)$: whether the TM can be driven from the configuration x to w in time $< 2^k$ and space s. For this we need for every z, to check $A(x, z, s, k-1)$ and $A(z, w, s, k-1)$, which takes space $t_k \le t_{k-1} + \|z\| + O(1)$. So, $t_k = O(sk) = O(s^2)$ [Savitch 70].

Search problems are games with P-time transition rules and one move duration. A great hierarchy of problems results from allowing more moves and/or other complexity bounds for transition rules.