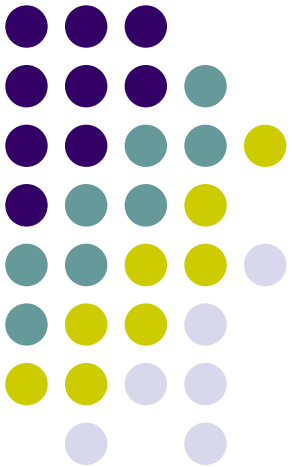
CS256
Applied Theory of Computation

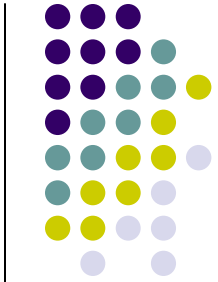
Introduction &

Complexity Classes I

John E Savage



Outline of the Course

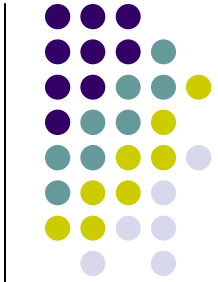


- Serial and parallel complexity classes – 3 lectures
- Approximation to NP-hard problems – 5 lectures
- Circuit complexity – 6 lectures
- Space-time tradeoffs – 4 lectures
- Memory hierarchies & I/O-space tradeoffs – 3 lects
- Parallel computation & classification – 4 lectures
- VLSI model, AT^2 tradeoffs, & algorithms – 4 lectures
- Quantum computation – 3 lectures

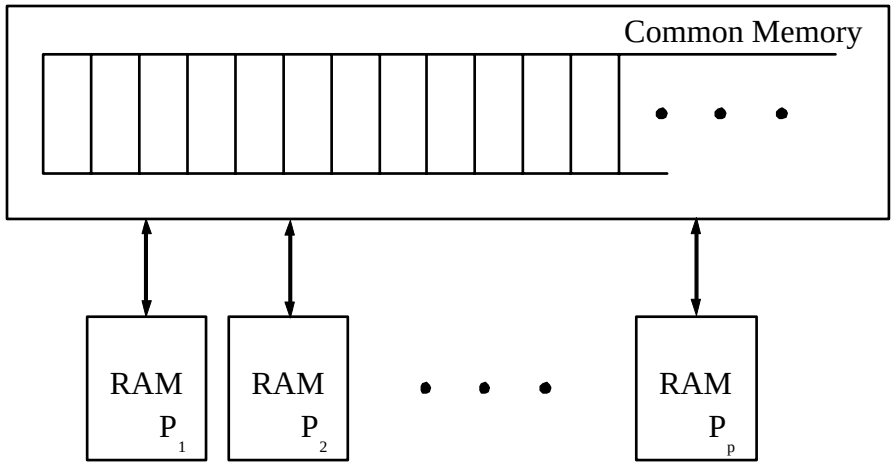
Background on Machine Models

- Memoryless serial and parallel machines
  - Logic & algebraic circuits (ring, field ops)
- Serial machines: RAM & TM
  - Memory hierarchies
- Parallel machines with memory
  - Fine- vs coarse-grained computers
  - PRAM - p RAMs with shared memory

Background on Machine Models (cont.)



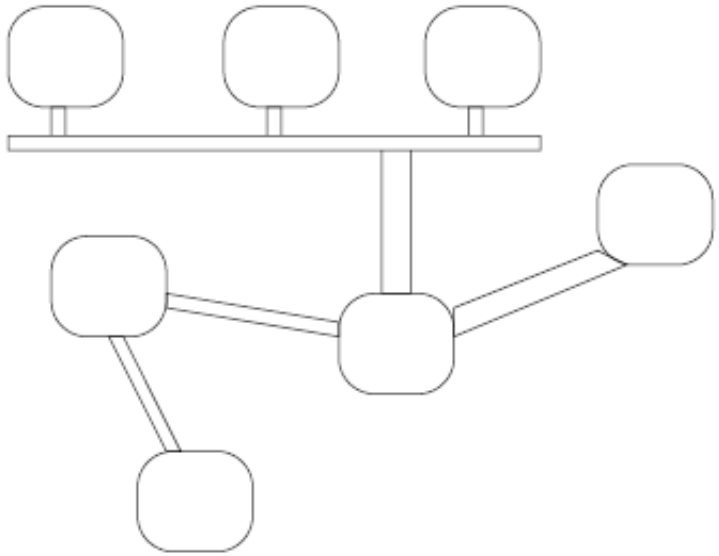
- Parallel machines with memory (cont.)
  - PRAM - p RAMs with shared memory



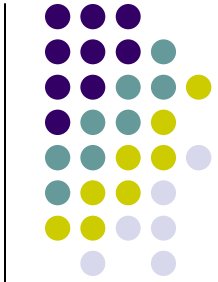
  - Loosely coupled network of computers
  - VLSI model

Background on Machine Models (cont.)

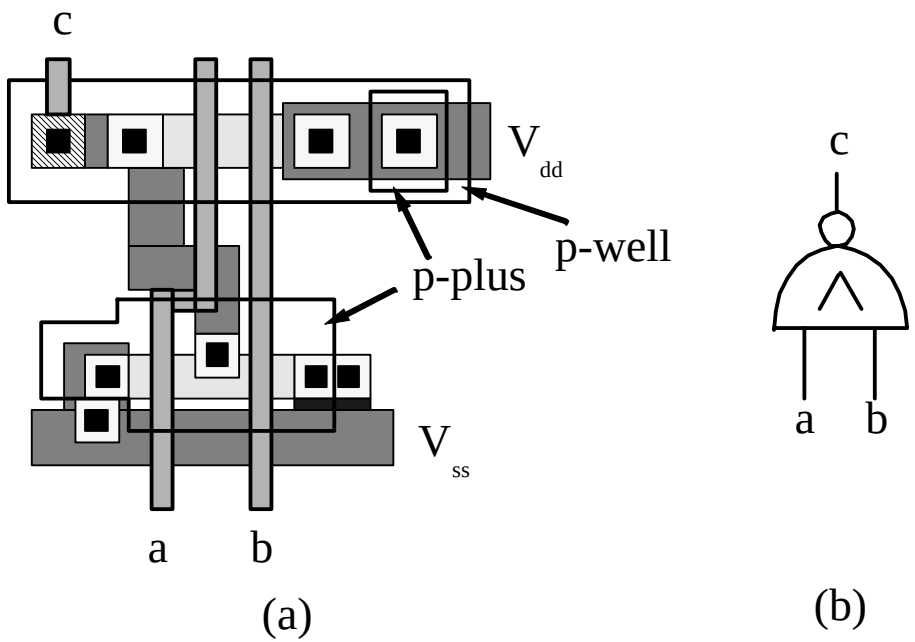
- Loosely coupled models



Background on Machine Models (cont.)



- VLSI model



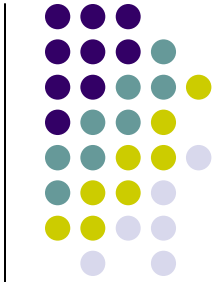
Performance Metrics

- Logical and algebraic circuits – circuit size & dept
- RAM and TM – Time & space
- Parallel machines – Time, no. processors, & space
- Memory hierarchies – I/O time vs primary storage
- Distributed computing
  - Time $T(n)$ to send length n message over single channel satisfies $T = l + nb$ where l is latency and b is bandwidth.

Complexity Classes I
Decision Problems and Classes

- Problems are classificed computational problems by their need for space and time
  - Problems usually defined by languages.
  - Use standard models of computation.
  - Use standard measures of space and time.
- A language L is defined as a subset of the set of strings over an alphabet, $L \subseteq \Sigma^*$
- The complement of the language L, denoted $\overline{L}$, is the set $\Sigma^* - L$

Decision Problems



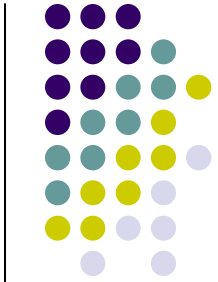
- Languages defined by **decision problems**.

  Satisfiability (SAT)

  Instance: A set of literals $X = \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$ and a sequence of clauses $C = (c_1, \ldots, c_m)$ where each c_i is a subset of X.

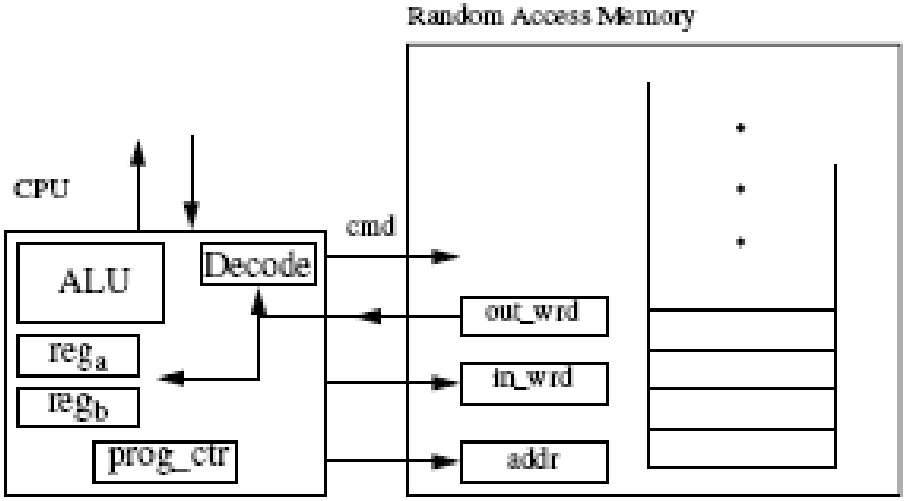
  Answer: "Yes" if for some assignment of values to Boolean variables $\{x_1, \ldots, x_n\}$ at least one literal in each clause has value 1.

Decision Problems



- Strings in satisfiability define a language.
- The complement of a decision problem *P*, denoted **co-***P*, is the set of "No" instances of a decision problem
- Note that **co-***P* is not $\overline{P}$ because **co-***P* $\cup$ *P* is the set of strings describing well-formed clauses, not all strings over Σ^*.

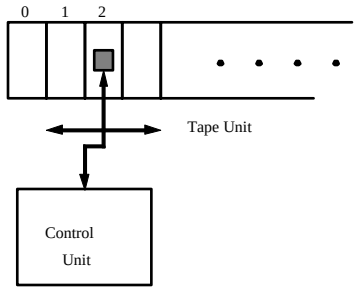
Standard Computational Model Random Access Machine



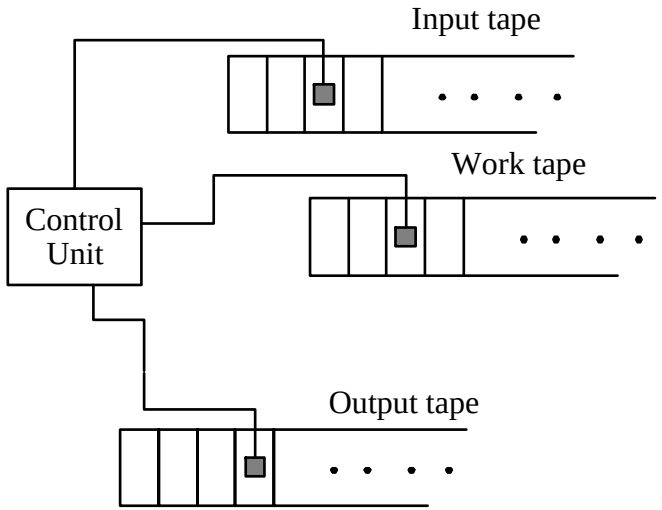
- Adds, subtracts, shift left or right one place, compare two words, perform Boolean vector AND, OR and NOT, do loads, stores, and conditional jumps and immediate and direct addressing.
- No multiplication or division allowed.
- If fixed-length addresses and fixed values are stored in memory initially, how big can addresses and values become?

Standard Computational Model Turing Machine

- The standard one-tape Turing Machine



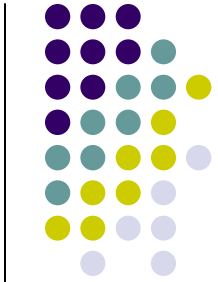
- The multi-tape TM
  - When space counts, input tape is read-only & space is number of cells used on work tape.



Standard Computational Models

- **Time** is number of steps executed. **Space** on the TM is number of tape cells used; on the RAM it is number of storage bits used.
- Time on TM and RAM can be shown to be within polynomial bounds of one another. Why is that?
  - Simulate T-step RAM on the TM.
  - Store RAM words as (*address, value*) pairs on the TM tape.
  - To overwrite a word, invalidate the previous pair without removing it and write a new one.
  - How many bits can each address have after T steps?
  - How long does it take to simulate a RAM step on the TM?
  - Does this imply $O(T^3)$ time simulation?

Resource Bounds



- Typical resource functions are logarithms, polynomials of logs, linear, polynomials, super-polynomials, and exponentials.
- We must avoid functions so complex that they cannot be computed in the time and/or space they are used to define.

  Definition *A function r: N → N is* ***proper*** *if it is non-decreasing & for a letter a there is a deterministic multi-tape TM M that on all inputs of length n in time O(n + r(n)) and space O(r(n)) writes the string* $a^{r(n)}$ *on one of its tapes and halts.*

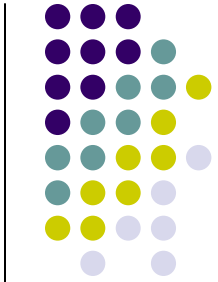
Proper Functions and Precise TMs

Theorem Let r(n) be a proper function with $r(n) \geq n$. Let M be a multi-tape DTM, NDTM, or oracle TM with k work tapes that computes a total function f in time or space r(n). Then there is a constant $K > 0$ and a precise Turing machine of the same type that computes f in time and space Kr(n).

Proper Functions and Precise TMs (cont.)

Proof Let M_p simulate in $K_1 r(n)$ steps the machine M_r that computes r(n) to write a string of length r(n) on a special enumeration tape as well as $K_1 r(n)$ special blank symbols on its work tapes. Since the number of tapes on M_p is finite, it uses kr(n) cells for some k. M_p alternates between writing on its tapes and reading from the enumeration tape. It continues to read once every two cycles from the enumeration tape after simulating M to insure that it uses exactly 2r(n) steps.

Resource Bounds



- If r(n) is proper, a DTM M_r exists that can be used to limit a computation on an input of length n to time O(n + r(n)) and space O(r(n))

Definition *A **precise multi-tape TM** (deterministic or not) has a proper function r(n) such that on every input of length n, it halts in precisely r(n) steps.*

Space and Time Complexity Classes

- Deterministic and nondeterministic space and time defined for DTMs & NDTMs using proper resource functions.
- TIME(r(n)) and SPACE(r(n)) are sets of languages accepted in r(n) deterministic time & space by DTMs
- NTIME(r(n)) and NSPACE(r(n)) are sets of langs. accepted in r(n) nondet. time & space by NDTMs
- The classes **P** and **NP**

$$\mathbf{P} = \cup_k \ \mathbf{TIME}(n^k)$$

$$\mathbf{NP} = \cup_k \ \mathbf{NTIME}(n^k)$$

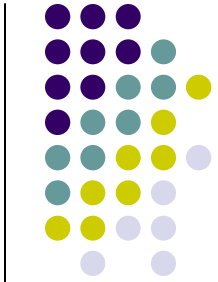
Space and Time Complexity Classes

- Language L_c is **NP**-complete language if it is in **NP** & for any other language L in **NP** there's a poly-time computable *translation function* t() such that **w** is in L if and only if t(**w**) is in L_c. That is, the decision problem for L can be settled in poly-time by L_c.
- Exponential complexity classes

$$\textbf{EXPTIME} = \cup_n \ \textbf{TIME}(2^n)$$

$$\textbf{NEXPTIME} = \cup_n \ \textbf{NTIME}(2^n)$$

Hierarchy Theorem



Time Hierarchy Theorem *If r(n) ≥ n is a proper resource function, then* ***TIME***(r(n)) *is strictly contained in* ***TIME***(r(n) *log* r(n)).

- Let r(n) and s(n) be proper resource functions. Then r(n) = o(s(n)) ("little oh") if for all K > 0 there exists an N_0 such that s(n) ≥ Kr(n) for n ≥ N_0.

Hierarchy and Gap Theorems

- **Space Hierarchy Theorem** *If r(n) and s(n) are proper resource functions and r(n) = o(s(n)), then* ***SPACE****(r(n)) is strictly contained in* ***SPACE****(s(n)).*
- **Gap Theorem** *There is a recursive function r(n):B* → B* such that* ***TIME****(r(n))=* ***TIME****(*$2^{r(n)}$*)*

Theorem: P ⊆ NP ⊆ EXPTIME ⊆ NEXPTIME but **P** is strictly contained in **EXPTIME.**