Lecture 2: The Turing Model

1 Review

In the last lecture, we reviewed the basic features of Complexity theory.

We noted that we can recover a form of Church's thesis on computational invariance provided if we introduce three key concepts that cuts across computational models.

- COMPUTATIONAL MODES (E.g., determinism, parallelism, probabilism, etc)
- COMPUTATIONAL RESOURCES (E.g., time, space and reversals)
- the notion of tractability parameterized by computational mode and computational resource. (E.g., NP=tractable time in nondeterministic mode)

Then the basic invariance principle says that these tractable classes are model independent.

In this lecture, we illustrate these concepts in two concrete computational models.

2 The Turing Model

Once we have agreed on the problem, we need to agree on the computational model. E.g., if P is the sorting problem, there are still many VERY different models (comparison tree, comparator network, tape models).

Our main model is the Turing Machine (TM). There are many variants of TM's, but we choose a main form for most of our work.

- Finitely many states Q. There is a start state and an accept state.
- $k+1$ tapes. Each tape is a doubly-infinite sequence of tape squares (cells), indexed by $\mathbb{Z}$.

  Tape 0 is the INPUT tape. Tapes 1 to k are WORK TAPES.
- Each cell can store one symbol from Σ.
- Each tape has a TAPE HEAD. Head i scans a single cell in tape i.
- ACTION: depending of scanned symbols and current state, can change the scanned symbol, state and move left/right/stationary.
- FORMALIZATION: A k-tape transition table is a finite subset δ of

  $$\Sigma_\infty^{k+1} \times Q_\infty \times \Sigma_\infty^k \times Q_\infty \times \{0, \pm 1\}^{k+1}.$$

  Each $(3k+4)$-tuple is an INSTRUCTION.

- CONVENTIONS:
  - the tapes are all blank except for the initial input word.
  - the input head scans the leftmost non-blank symbol of input word.
  - the input head never scan past the first blank on the input tape.
  - the blank word is never written.
  - there is no transition out of q_a, q_r and no transitions into q_0.

- COMPUTATION:

  –A configuration of δ is

$$C = \langle q, w_0, n_0, w_1, n_1, \ldots, w_k, n_k \rangle$$

  –Initial configuration on an input $C_0(x)$.

  –A configuration is accepting/rejecting/terminal.

  –The successor relation $C \vdash_\delta C'$ defined in the natural way.

  –Computation path: begins with an initial configuration, follows the $\vdash$ relation, and is either infinite or ends at a terminal configuration. Say the path is accepting/rejecting/terminal, depending on the last configuration (if exists).

  –Computation tree.

- COMPUTATIONAL MODES:

  We say δ is deterministic if each configuration has a unique successor.

  Nondeterministic Acceptance Rule: word x is accepted iff there is an accepting computation path from $C_o(x)$.

  NONDETERMINISTIC MACHINE: M is a δ with the nondeterministic acceptance rule.

  DETERMINISTIC MACHINE: a nondeterministic M whose δ happens to be deterministic.

  Language accepted by M is $L(M)$.

  PROBABILISTIC MODE:

- COMPLEXITY:

  Let C be a configuration and π a computation path.

  Define $TIME(\pi)$.

  Define $SPACE(C)$ and and $SPACE(\pi)$.

  Define $REV(\pi)$.

  TIME: Let $AcceptTime_M(x)$ be the minimum length of an accepting computation path on x. This is ∞ if x is not accepted. Let $AcceptTime_M(n)$ be the maximum over all $AcceptTime_M(x)$, $|x| = n$ (may be ∞). If f is a complexity function, we say M accepts in time f if $AcceptTime_M(n) \leq f(\text{ev.})$.

SPACE and REVERSAL: Similarly, define $AcceptSpace_M(x)$ and $AcceptRev_M(x)$ to be the minimum space/rev of an accepting computation path.

RUNNING TIME: $RunningTime_M(x)$ is the maximum length of ALL computation paths on input x. (Regardless of whether x is accepted or not)

COMPLEXITY CLASSES: Let F be a family of complexity functions. E.g., $F = O(n), O(\log n), n^{O(1)}$, etc.

$NTIME(F)$ to be languages accept in time f for some $f \in F$.

Canonical Classes: $DLOG = DTIME(\log n)$, NLOG, $P = DTIME(n^{O(1)}$, $NP = NTIME(n^{O(1)}$, $PSPACE = DSPACE(n^{O(1)})$, etc.

SIMULTANEOUS COMPLEXITY BOUNDS:

$DTIME - SPACE(t, s)$

3 THE POINTER MODEL

Briefly: each machine has a finite set of states and a finite set of symbols Δ.

A Δ-structure is a digraph G in each each node has outdegree $|\Delta|$, and each edge is labeled with some $a \in \Delta$. Moreover, no two outgoing edges from a node has the same label.

One node is the CENTER. Now, EVERY word $w \in \Delta^*$ designates a unique node in G, with the empty word designating the CENTER.

An Δ-instruction has the form

(i) $w \leftarrow w'$

(ii) $w \leftarrow new$

(iii) if $w = w'$ then goto L (i.e., the Lth instruction)

A pointer machine is a finite sequence of such instructions.

INPUT CONVENTION:

Let a symbol $c \in \Delta$ be the "content" label. Let some of the nodes of Δ be interpreted as the symbols of a set Σ. A node u contains symbol $\sigma \in \Sigma$ if the c-edge from u points to the σ node.