Restricted Turing Machines

Zbyněk Křivka

krivka@fit.vutbr.cz

based on

Szepietowski, A.: Turing Machines with Sublogarithmic Space. Springer, 1994 (Chapters 1 through 5)

Formal Model Research Group

Faculty of Information Technology,

Brno University of Technology, Czech Republic

Contents

Turing Machine

- All recursively enumerable functions
- All algorithmically described languages
- Type-0 grammars
- …
- Almost all problems are undecidable and many are untractable (not **P with small *n***)

    ⇒ restrict space of TM

    ⇒ reduce power but better tractability

Space-bounded Turing Machines

- Assume: 2-way, read-only input, read-write work tape
- Complexity measure: Space (Strongly, Weakly)
- Log-space: Model independent
- Constant-space: Power?
- Sublog-space: How small bit of information improves finite automaton?
- Differences of log vs sublog bounded-space TMs:
    - Depends on the machines models & modes of space complexity (but lower bound same for many models).
    - $L(n) \geq \log n$ closed under catenation, not $L(n) = o(\log n)$; Below $\log n$, no unbounded non-decreasing function is fully space constructible.
    - More sophisticated proof techniques.

Turing Machine

Turing Machine (**TM**) is a sixtuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F), \text{ where}$$

- Q—finite set of *states*,
- Σ—an *input alphabet*
- Γ—a *tape alphabet* with the *blank symbol* $\square \in \Gamma$,
- $q_0 \in Q$—*start state*,
- $F \subseteq Q$—a *set of accepting states*,
- δ: $Q \times \Sigma \cup \{\blacktriangleright\blacktriangleleft\} \times \Gamma \rightarrow Power((\Gamma - \{\square\}) \times Q \times \{R, N, L\}^2)$— *the transition function* describing rules of the form

$$\boldsymbol{a p t_r \rightarrow t_w A_{wt}\, q A_{in}}$$

where $p, q \in Q$, $a \in \Sigma$, $t_r, t_w \in \Gamma$, $A_{wt}, A_{in} \in \{R, N, L\}$.

Deterministic Turing Machine

Deterministic Turing Machine **(DTM)** M is a **TM** where

δ: $Q\times\Sigma\cup\{\blacktriangleright\blacktriangleleft\}\times\Gamma \rightarrow (\Gamma-\{\square\})\times Q\times\{R, N, L\}^2$

In other words:

For every $(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{t_r})$,

there is at most one $(\boldsymbol{t_w}, \boldsymbol{q}, \boldsymbol{A_{wt}}, \boldsymbol{A_{in}})$ in $\delta(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{t_r})$,

where $\boldsymbol{p}, \boldsymbol{q}\in Q$, $\boldsymbol{a}\in\Sigma$, $\boldsymbol{t_r}, \boldsymbol{t_w}\in\Gamma$, $\boldsymbol{A_{wt}}, \boldsymbol{A_{in}}\in\{R, N, L\}$.

- One-way TM: input head cannot move to the left

Configuration

Configuration of M on an input w:

$$(\blacktriangleright w_1 \boldsymbol{q} a w_2 \blacktriangleleft, \boldsymbol{x_1 q x_2}) \text{ or } (i, \boldsymbol{x_1 q x_2})$$

where $\boldsymbol{q} \in Q$, $w = w_1 a w_2 \in \Sigma^*$, $0 \leq i \leq |w|+1$

$$x = x_1 x_2 \in (\Gamma - \{\square\})^*$$

In addition: $w[0] = \blacktriangleright$, $w[|w|+1] = \blacktriangleleft$, $x[|x|+1] = \square$

- input tape CANNOT change
- just possition of input head is sufficient

Initial Configuration:

$$(\blacktriangleright \boldsymbol{q_0} w \blacktriangleleft, \boldsymbol{q_0}\varepsilon) \text{ or simply } (1, \boldsymbol{q_0}\varepsilon)$$

Computation

Computation step:

$$(i,\ x_1 p x_2) \Rightarrow (i',\ x_1' q x_2')\ [a p t_r \rightarrow t_w A_{wt} q A_{in}]$$

where $|x_1| = j - 1$, M does:

1. write t_w on $x[j]$;
2. do action A_{wt} on the work tape;
3. do action A_{in} on the input tape;
4. change the current state to q.

M cannot enter more than c configurations, where

$$c = |Q| \cdot (|w|+2) \cdot |x| \cdot |\Gamma|^{|x|}$$

Internal Configuration

- *final configuration* = no computation step possible
- *accepting configuration* = final configuration with accepting state
- *computation* = finite or infine sequence of configurations

---

***Internal Configuration of M*:**

$$x1qx2$$

where $q \in Q$,

$x = x_1x_2 \in (\Gamma - \{\square\})^*$,

and j is the *position of the work head*, $1 \leq j \leq |x|+1$.

Upper bound for the number of all internal configurations: $d^{|x|} = |Q| \cdot |x| \cdot |\Gamma|^{|x|}$

Space Complexity

- the maximal space used by configurations of the computation
- recall that every visited cell is non-blank

***L*(*n*)** be a function on natural number. Let $w = |n|$.

Strongly L*(*n*) *space-bounded TM:

if no accessible configuration on **any input** *w* uses more than $L(n)$ cells on the work tape.

Weakly L(*n*) *space-bounded TM*:

If for every accepted input *w*, at least one accepting computation uses at most $L(n)$ space.

Middle L(*n*) *space-bounded TM*:

If no accessible configuration on any accepted input *w* uses more than $L(n)$ space.

Space Complexity Classes

***DSPACE*[*L*(*n*)], *NSPACE*[*L*(*n*)] –** class of languages accepted by **deterministic** and **nondeterministic** TM, respectively.

• Add prefix *strong*, *weak*, or *middle*, if needed; otherwise the results holds for all types of the definition.

Notation: (In literature: = corresponds to $\in$)

$f(n) = O(g(n))$ if there exists $c > 0$, s. t. $f(n) \leq cg(n)$.

$f(n) << g(n)$ if $\liminf_{n\to\infty}(f(n)/g(n)) = 0$.

$f(n) = o(g(n))$ if $\lim_{n\to\infty}(f(n)/g(n)) = 0$.

• The logarithm function **log *n*** is in **base 2**.

TM with Logarithmic Space

- TM with logarithmic or greater space can
  - store on the work tape numbers up to the size of the input;
  - remember any position on the input tape.
- Eg. *GAP* (Graph accessibility problem) language

Example 1: Primes

$$\{a^n : n \text{ is prime}\}$$

- counts the letters of an input
- stores the number in binary on the work tape
- checks one by one for each $1 < k < n$, whether k divides n
- accepts if no k divides n.

Example 2: Reflection

$$\{ww^R : w \in \{0,1\}^*\}$$

- compare the first letter with the last one
- compare the second with the last by one
- …
- just track the current position in binary on the work tape

Pebble Automata

A k-pebble finite automaton **(*k*-PA)**:

- *two-way read-only input tape* (no work tape),
- *k pebbles* which can be placed on and removed from the input tape (bound to the concrete cell),
- *finite set of rules* of the form

$$\boldsymbol{qaP \rightarrow q\{\mathrm{N, R, L}\}\{\mathrm{drop, take}\}}$$

where $p, q \in Q$, $a \in \Sigma$, P is a set of pebbles on the current cell.

Example 3: Reflection in PA

$$\{ww^R : w \in \{0,1\}^*\}$$

- How much pebbles do we need?
- What is the power of 1-pebble automata?
- What is the power of k-pebble automata?

Power of Pebble Automata

Theorem: *k*-PA = *log-space-TM*, with *k* depending on the number of work tape symbols.

Proof:

See page 16-18.

GAP Language

GAP language consists of encoded directed graphs which have a path from the first to last vertex.

A directed graph $G = (V, E)$, where $E \subseteq V \times V$.

Encoded as

$$**a_1*a_{1,1}*a_{1,2}\ldots a_{1,i_1}**a_2*a_{2,1}*a_{2,2}\ldots a_{2,i_2}**\ldots$$

Thus, lists of vertices reachable from the list head.

NSPACE(log n) Complete Languages

Lemma: *GAP* $\in$ *NSPACE*(log *n*).

Proof: page 18

Lemma: *GAP* is *NSPACE*(log *n*) complete.

Proof: page 19

NSPACE(log *n*) Complete Languages

Lemma: If A_1 is log-space reducible to A_2 and $A_2 \in$ ***DSPACE*(log *n*)** then $A_1 \in$ ***DSPACE*(log *n*)**.

Proof: page 19

Theorem: ***GAP*** $\in$ ***DSPACE*(log *n*)** iff ***NSPACE*(log *n*)** = ***DSPACE*(log *n*)** .

TM with Sublogarithmic Space

- **Constant space-bounded** TM accepts regular languages (Hopcroft and Ullman 1979)
- ***L*(*n*) << log *n***: e.g. Primes (even primes are trivial)
  - a little tricky definition of <<
- The first real non-regular **sublog-space** language
  - Stearns et al. 1965

$$w_k = b_0 \# b_1 \# \dots \# b_k$$

where b_i is binary description of the number i.

Example 4: Numbers

$$w_k = b_0 \# b_1 \# \ldots \# b_k$$

- compare b_0 with b_1
- compare b_1 with b_2
- …
- just track the current position in binary representation of b_i

$$L(n) = \lfloor \log \lfloor \log k \rfloor \rfloor + 1$$

Example 5: What is log log *n*?

n	**log *n***	**log log *n***
2	1	0
4	2	1
16	4	2
256	8	3
65536	16	4
4294967296	32	5
1,84467E+19	64	6
3,40282E+38	128	7
1,15792E+77	256	8

Example 6: Nonequivalence

$$A = \{a^k b^m : k \neq m\}$$

- A is non-regular
- $A \in$ ***weak-DSPACE*[log log *n*]**
- $A \in$ ***weak-one-way-NSPACE*[log log *n*]**
- Trick (For proof see pages 22 through 24):
  - M guesses j such that $k \neq m$ (mod j) and
  - $j < c \log |k + m|$.
- $A \notin$ ***strong-NSPACE*[sublog *n*]**

What are the lower bounds?

Lower Bounds for Accepting Non-regular Languages

- **Gap theorems**: no use of constant-bounded or less than (d log log n) bounded-space to get non-regular languages.
- Lower bound for weakly space-bounded one-way TMs is log n for deterministic and log log n for nondeterministic (Alberts 1985).
- TMs with 2-dimensional inputs can be space-bounded by $\log^* n$ or $\log^{(k)} n$

Lower Bounds for Two-way TMs

Theorem: Let M be a **weakly** $L(n)$ space-bounded **deterministic or non-deterministic TM**. Then either:

- $\boldsymbol{L(n) \geq c \log \log n}$ with $c > 0$ and inf. many n,

or

- M accepts a **regular language** and space used by M is **bounded by a constant**.

Proof: *k*-equivalent suffixes

***C*(*k*, *M*) – set of all *k* space-bounded (s-b) internal cfgs of *M*.**

$(\beta_1, \beta_2) \in$ ***P*(*k*, *M*, *w*)** iff there is *k* s-b computation of *M* starting in β_1 **at *w*[1] reaches** β_2 **just after it leaves *w* to the left.**

$(\beta) \in$ ***Q*(*k*, *M*, *w*)** iff there is *k* s-b **accepting** computation of *M* starting in β **at leftmost letter of *w* accepts** without leaving *w*.

***k*-equivalent *u* and *v*,** $u,v \in \Sigma^*$:

$$u \equiv_k v \text{ iff}$$

$$P(k, M, u) = P(k, M, v) \text{ and } Q(k, M, u) = Q(k, M, v).$$

Intuitively: *M* cannot distinguish *k*-equivalent suffixes when using *k* space.

Proof: Auxiliary Lemma

Lemma: Let $u, v, x \in \Sigma^*$, and $\boldsymbol{u \equiv_k v}$. Then:
There is k s-b accepting computation of M on xu **iff**
there is k s-b accepting computation of M on xv.

Proof (see page 29):

- Study crossing x-u boundary: divide computation into segments $\alpha_1, \ldots, \alpha_j$, where α_i satisfy (a) with odd i enters x, and (b) with even i enters u.
- From assumption $u \equiv_k v$, for α_i and even i, there is corresponding δ_i (by analogy for even j).
- For even i, replace α_i by δ_i and we obtain computation of M on xv.

Proof of Theorem

Part: $\boldsymbol{L(n) \geq c \log \log n}$ with $c > 0$ and inf. many n

Proof (page 29): Part 1) Contradition of ($w_i \equiv_k w_j$ and $w_i \equiv_{k\text{-}1} w_j$)

- Suppose no constant upper s-b $\Rightarrow$ inf. many k and w.
- $w = a_1 a_2 \ldots a_n$ the **shortest** input accepted **in *k* space**.
- $w_i = a_i \ldots a_n$, $w_j = a_j \ldots a_n$, $i < j$.
- Suppose $w_i \equiv_k w_j$ and $w_i \equiv_{k\text{-}1} w_j$ for some $1 \leq i < j \leq n$.
- From lemma, there is k s-b accepting computation on $w' = a_1 a_2 \ldots a_{i\text{-}1} a_j \ldots a_n$.
- As w is the shortest input & $w_i \equiv_{k\text{-}1} w_j$, w' **cannot use less than** k; otherwise w also accepted in less space.

Proof of Theorem

Part: $\boldsymbol{L(n) \geq c \log \log n}$ with $c > 0$ and inf. many n

Proof (Part 2, page 29): Recall: $\equiv_k$ is an equivalence relation; Let $c = d^{|x|} = |Q| \cdot |x| \cdot |\Gamma|^{|x|}$, where $|x| = k$ and $|w| = n$.

- The number of possible equivalence classes $\equiv_k$

  $f_k =$ „$\#\boldsymbol{P}(k, M, w) \cdot \#\boldsymbol{Q}(k, M, w)$" $= 2^{(c \cdot c)} \cdot 2^{c} \leq 4^{(c \cdot c)}$

- Since two diff. suffixes w_i, w_j cannot belong to the same equivalence classes of $\equiv_k$ and $\equiv_{k-1}$, it requires that

$$(4^{(c \cdot c)})^2 \geq f_k \cdot f_k = (4^{(c \cdot c)}) \cdot (4^{(c' \cdot c')}) \geq n$$

$$\log (4^{(c \cdot c)})^2 \geq \log n$$

$$\log (2\, d^{|x|} \cdot d^{|x|}) \geq \log \log n$$

$$\boldsymbol{k \geq h \log \log n}.$$

- Hence, $L(n) \geq h \log \log n$ for const. $h > 0$ & infinite many n.

Conclusion

- Open problems:
  - deterministic vs non-deterministic TMs

- **Read the book!**

Discussion