Machine Models for Stream-Based Processing of External Memory Data

Nicole Schweikardt

Humboldt-University Berlin

Workshop on Algorithms for Data Streams

IIT Kanpur

18 – 20 December 2006

Overview

Overview

A model based on Turing machines
- The ST-model
- One external memory tape
- Several external memory tapes
- Future Tasks

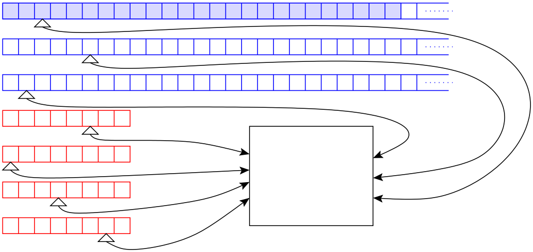
A model for database query processing: Finite Cursor Machines

Goal: Machine Model for . . .

- fast & small internal memory **vs.** huge & slow external memory
- external memory: random access **vs.** sequential scans
- several external memory devices

- machine model and complexity classes that measure costs caused by external memory accesses
- lower bounds for particular problems

Turing Machine Model



multi-tape Turing machine with

- *t* “long” tapes (that represent *t* external memory devices)
  … limited access
- some “short” tapes (that represent internal memory)
  … limited size

Input on the first external memory tape.
If necessary: Output on the *t*-th external memory tape.

Head Reversals

- When the external memory tape models a hard disk or a data stream, it should be read only in one direction (from left to right).
- For our *lower bounds* we still allow head reversals on the external memory tape. (This makes our lower bound results only stronger.)
- Allowing head reversals, we can ignore random access, because each "random access jump" can be simulated by at most 2 head reversals.

(r,s,t)-Bounded Turing Machines

Let $r:\mathbb{N}\to\mathbb{N}$, $s:\mathbb{N}\to\mathbb{N}$, $t\in\mathbb{N}$.

A (nondeterministic) Turing machine is called (r,s,t)-bounded if it has

- at most t external memory tapes,
- internal memory tapes of total length $\leqslant s(N)$,
- less than $r(N)$ head reversals on the external memory tapes

(where $N =$ input length). $(r(N) \approx$ # sequential scans of external memory)

- $\mathrm{ST}(r,s,t)$ = class of all problems solvable by deterministic (r,s,t)-bounded TMs
- $\mathrm{NST}(r,s,t)$ = class of all decision problems solvable by nondeterministic (r,s,t)-bounded TMs
- $\mathrm{RST}(r,s,t)$ = class of all decision problems solvable by randomized (r,s,t)-bounded TMs with the following acceptance criterion:

accept each "yes"-instance with probability > 0.5,
reject each "no"-instance with probability 1.

(r, s, t)-Bounded Turing Machines

Let $r : \mathbb{N} \to \mathbb{N}$, $s : \mathbb{N} \to \mathbb{N}$, $t \in \mathbb{N}$.

A (nondeterministic) Turing machine is called (r, s, t)-bounded if it has

- at most t external memory tapes,
- internal memory tapes of total length $\leqslant s(N)$,
- less than $r(N)$ head reversals on the external memory tapes

(where $N =$ input length). ($r(N) \approx$ # sequential scans of external memory)

- $\mathrm{ST}(r, s, t)$ = class of all problems solvable by deterministic (r, s, t)-bounded TMs
- $\mathrm{NST}(r, s, t)$ = class of all decision problems solvable by nondeterministic (r, s, t)-bounded TMs
- $\mathrm{RST}(r, s, t)$ = class of all decision problems solvable by randomized (r, s, t)-bounded TMs with the following acceptance criterion:

accept each "yes"-instance with probability > 0.5,
reject each "no"-instance with probability 1.

(r, s, t)-Bounded Turing Machines

Let $r : \mathbb{N} \to \mathbb{N}$, $s : \mathbb{N} \to \mathbb{N}$, $t \in \mathbb{N}$.

A (nondeterministic) Turing machine is called (r, s, t)-bounded if it has

- at most t external memory tapes,
- internal memory tapes of total length $\leqslant s(N)$,
- less than $r(N)$ head reversals on the external memory tapes

(where $N =$ input length). ($r(N) \approx$ # sequential scans of external memory)

- $\mathrm{ST}(r, s, t)$ = class of all problems solvable by deterministic (r, s, t)-bounded TMs
- $\mathrm{NST}(r, s, t)$ = class of all decision problems solvable by nondeterministic (r, s, t)-bounded TMs
- $\mathrm{RST}(r, s, t)$ = class of all decision problems solvable by randomized (r, s, t)-bounded TMs with the following acceptance criterion:

accept each "yes"-instance with probability > 0.5,
reject each "no"-instance with probability 1.

Special Cases

$\mathrm{ST}(1, s, t)$:

- input is a data stream,
- only internal memory available for the computation,
- output consists of up to $t-1$ data streams

$\mathrm{ST}(r, s, 1)$:

- one hard disk is available,
- input and output at this hard disk,
- the hard disk may be used throughout the computation,
- $\leqslant r(N)$ sequential scans of the hard disk,
- internal memory of size $\leqslant s(N)$.

In particular, $\mathrm{ST}(r, s, 1)$ comprises the W-Stream model of Demetrescu, Finocchi, Ribichini (SODA'06)

Special Cases

$\mathrm{ST}(1, s, t)$:

- input is a data stream,
- only internal memory available for the computation,
- output consists of up to $t-1$ data streams

$\mathrm{ST}(r, s, 1)$:

- one hard disk is available,
- input and output at this hard disk,
- the hard disk may be used throughout the computation,
- $\leqslant r(N)$ sequential scans of the hard disk,
- internal memory of size $\leqslant s(N)$.

In particular, $\mathrm{ST}(r, s, 1)$ comprises the W-Stream model of Demetrescu, Finocchi, Ribichini (SODA'06)

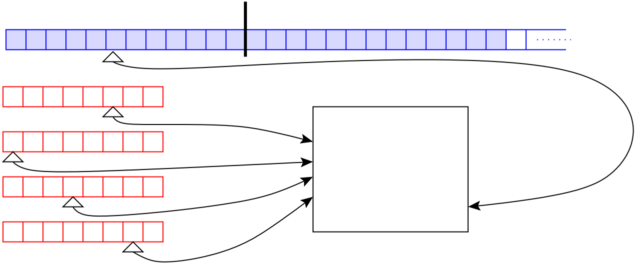
Overview

A model based on Turing machines

- The ST-model
- One external memory tape
- Several external memory tapes
- Future Tasks

A model for database query processing: Finite Cursor Machines

An Easy Observation



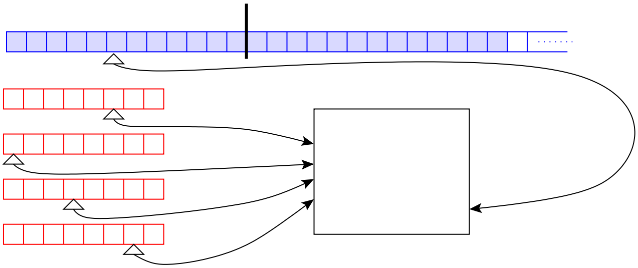
Fact:

During an $(r, s, 1)$-bounded computation, only $O(r(N) \cdot s(N))$ bits can be communicated between the first and the second half of the external memory tape.

Consequence:

Lower bounds on communication complexity lead to lower bounds for the $\mathrm{ST}(\cdot, \cdot, 1)$ classes.

An Easy Observation



Fact:

During an $(r, s, 1)$-bounded computation, only $O\big(r(N) \cdot s(N)\big)$ bits can be communicated between the first and the second half of the external memory tape.

Consequence:

Lower bounds on communication complexity lead to lower bounds for the $\mathrm{ST}(\cdot, \cdot, 1)$ classes.

Multiset Equality

MULTISET-EQUALITY *Input length:* $N = O(m \cdot n)$ *Bits*

Input: Two multisets $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_m\}$ of Bit-strings x_i, y_j (w.l.o.g. they all have the same length n)

Question: Is $\{x_1, \ldots, x_m\} = \{y_1, \ldots, y_m\}$?

Theorem:

MULTISET-EQUALITY $\in \mathrm{ST}(r, s, 1) \iff r(N) \cdot s(N) \in \Omega(N)$

Proof:

"$\Longrightarrow$": use communication complexity lower bound for *set-equality*

"$\Longleftarrow$": show that sorting is possible when $r(N) \cdot s(N) \in \Omega(N)$

Theorem:

MULTISET-EQUALILTY $\in$ co-RST$(2, O(\log N), 1)$

Proof: standard fingerprinting techniques $\rightsquigarrow$ data stream algorithm that always accepts all "yes"-instances and that rejects "no"-instances with high probability.

Multiset Equality

MULTISET-EQUALITY	*Input length:* $N = O(m{\cdot}n)$ *Bits*
Input:	Two multisets $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_m\}$ of Bit-strings x_i, y_j (w.l.o.g. they all have the same length n)
Question:	Is $\{x_1, \ldots, x_m\} = \{y_1, \ldots, y_m\}$?

Theorem:

MULTISET-EQUALITY $\in \mathrm{ST}(r, s, 1) \iff r(N) \cdot s(N) \in \Omega(N)$

Proof:

"$\Longrightarrow$": use communication complexity lower bound for *set-equality*

"$\Longleftarrow$": show that sorting is possible when $r(N) \cdot s(N) \in \Omega(N)$

Theorem:

MULTISET-EQUALILTY $\in$ co-RST$(2, O(\log N), 1)$

Proof: standard fingerprinting techniques $\rightsquigarrow$ data stream algorithm that always accepts all "yes"-instances and that rejects "no"-instances with high probability.

Multiset Equality

MULTISET-EQUALITY *Input length:* $N = O(m{\cdot}n)$ *Bits*

Input: Two multisets $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_m\}$ of Bit-strings x_i, y_j (w.l.o.g. they all have the same length n)

Question: Is $\{x_1, \ldots, x_m\} = \{y_1, \ldots, y_m\}$?

Theorem:

MULTISET-EQUALITY $\in \mathrm{ST}(r, s, 1) \iff r(N) \cdot s(N) \in \Omega(N)$

Proof:

"$\Longrightarrow$": use communication complexity lower bound for *set-equality*

"$\Longleftarrow$": show that sorting is possible when $r(N) \cdot s(N) \in \Omega(N)$

Theorem:

MULTISET-EQUALILTY $\in$ co-RST$(2, O(\log N), 1)$

Proof: standard fingerprinting techniques $\rightsquigarrow$ data stream algorithm that always accepts all "yes"-instances and that rejects "no"-instances with high probability.

Some Further Results for $\mathrm{ST}(r, s, 1)$

XML query processing:

Q-FILTERING (for a Core XPath query Q)

Input: XML-Document D

Question: Is $Q(D) \neq \emptyset$, i.e. does Q select at least one node in D ?

Theorem: *(Grohe, Koch, S., ICALP'05)*

(a) *For every Core XPath query Q we have:* $Q\text{-FILTERING} \in \mathrm{ST}(1, O(\textit{Höhe}(D)))$

(b) *There is a Core XPath query Q such that for all r, s with* $r(D) \cdot s(D) \in o(\textit{height}(D))$ *we have:* $Q\text{-FILTERING} \notin \mathrm{ST}(r, s)$.

A hierarchy w.r.t. the number of head reversals:

(similar result also for $\mathrm{RST}(\cdot, \cdot, 1)$)

Theorem: *(Hernich, S., 2006)*

For every logspace-computable function r with $r(N) \in o\left(\frac{N}{\log^2 N}\right)$ *and for every class S of functions with* $O(\log N) \subseteq S \subseteq o\left(\frac{N}{r(N) \cdot \log N}\right)$ *we have:*

$\mathrm{ST}(r(N), S, 1) \subsetneq \mathrm{ST}(r(N){+}1, S, 1)$

Some Further Results for ST$(r, s, 1)$

XML query processing:

Q-FILTERING (for a Core XPath query Q)

Input: XML-Document D

Question: Is $Q(D) \neq \emptyset$, i.e. does Q select at least one node in D ?

Theorem: *(Grohe, Koch, S., ICALP'05)*

(a) *For every Core XPath query Q we have:* Q-FILTERING $\in$ ST$(1, O(Höhe(D)))$

(b) *There is a Core XPath query Q such that for all r, s with* $r(D) \cdot s(D) \in o\big(height(D)\big)$ *we have:* Q-FILTERING $\notin$ ST(r, s).

A hierarchy w.r.t. the number of head reversals:

(similar result also for RST$(\cdot, \cdot, 1)$)

Theorem: *(Hernich, S., 2006)*

For every logspace-computable function r with $r(N) \in o\big(\frac{N}{\log^2 N}\big)$ *and*

for every class S of functions with $O(\log N) \subseteq S \subseteq o\Big(\frac{N}{r(N) \cdot \log N}\Big)$ *we have:*

$\text{ST}(r(N), S, 1) \subsetneqq \text{ST}(r(N){+}1, S, 1)$

Some Further Results for ST$(r, s, 1)$

XML query processing:

Q-FILTERING (for a Core XPath query Q)

Input: XML-Document D

Question: Is $Q(D) \neq \emptyset$, i.e. does Q select at least one node in D ?

Theorem: *(Grohe, Koch, S., ICALP'05)*

(a) *For every Core XPath query Q we have:* Q-FILTERING $\in \mathrm{ST}(1, O(\mathit{Höhe}(D)))$

(b) *There is a Core XPath query Q such that for all r, s with* $r(D) \cdot s(D) \in o\big(\mathit{height}(D)\big)$ *we have:* Q-FILTERING $\notin \mathrm{ST}(r, s)$.

A hierarchy w.r.t. the number of head reversals:

(similar result also for $\mathrm{RST}(\cdot, \cdot, 1)$)

Theorem: *(Hernich, S., 2006)*

For every logspace-computable function r with $r(N) \in o\big(\frac{N}{\log^2 N}\big)$ *and*

for every class S of functions with $O(\log N) \subseteq S \subseteq o\Big(\frac{N}{r(N) \cdot \log N}\Big)$ *we have:*

$\mathrm{ST}(r(N), S, 1) \subsetneq \mathrm{ST}(r(N)+1, S, 1)$

Overview

A model based on Turing machines

- The ST-model
- One external memory tape
- Several external memory tapes
- Future Tasks

A model for database query processing: Finite Cursor Machines

Situation with $t \geqslant 2$ EM-Tapes

Problem:

An additional external memory tape can be used to move around large parts of the input (with just 2 head reversals).

$\rightsquigarrow$ communication complexity does not help to prove lower bounds.

Example: The SORTING PROBLEM:

SORT $\qquad$ *input length* $N = m \cdot (n+1)$

Input: Bit-strings $x_1, \ldots, x_m \in \{0,1\}^n$ (for arbitrary m, n)

Output: $x_1, \ldots, x_m$ sorted in ascending order

Recall: SORT $\in \mathrm{ST}(r, s, 1) \iff r(N) \cdot s(N) \in \Omega(N)$.

Theorem (Chen, Yap, 1991): SORT $\in \mathrm{ST}(O(\log N), O(1), 2)$

Proof method: Merge-Sort.

Situation with $t \geqslant 2$ EM-Tapes

Problem:

An additional external memory tape can be used to move around large parts of the input (with just 2 head reversals).

$\rightsquigarrow$ communication complexity does not help to prove lower bounds.

Example: The SORTING PROBLEM:

SORT	*input length* $N = m \cdot (n+1)$
Input:	Bit-strings $x_1, \ldots, x_m \in \{0,1\}^n$ (for arbitrary m, n)
Output:	$x_1, \ldots, x_m$ sorted in ascending order

Recall: $\text{SORT} \in \text{ST}(r, s, 1) \iff r(N) \cdot s(N) \in \Omega(N)$.

Theorem (Chen, Yap, 1991): $\text{SORT} \in \text{ST}(O(\log N), O(1), 2)$

Proof method: Merge-Sort.

Bounds for Sorting with $t \geqslant 2$ EM-Tapes

Proposition:

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible.
Then there is an r with $r(N) \in O(\log \frac{N}{s(N)})$ such that $\text{SORT} \in \text{ST}(r(N), s(N), 2)$

Proof: Refine Chen and Yap's implementation of Merge-Sort

Main Theorem: *(Grohe, Hernich, S., 2006)*

Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(N) \in o(N)$.
For every r with $r(N) \in o(\log \frac{N}{s(N)})$ we have $\text{SORT} \notin \text{ST}(r(N), s(N), O(1))$
and $\text{MULTISET-EQUALITY} \notin \text{RST}(r(N), s(N), O(1))$

Corollary:

(a) $\text{SORT} \notin \text{ST}(o(\log\log N), O(\frac{N}{\log N}), O(1))$; $\text{SORT} \in \text{ST}(O(\log\log N), O(\frac{N}{\log N}), O(1))$

(b) *For every ε with* $0 < \varepsilon < 1$ *we have* $\text{SORT} \notin \text{ST}(o(\log N), N^{1-\varepsilon}, O(1))$
and $\text{MULTISET-EQUALITY} \notin \text{RST}(o(\log N), N^{1-\varepsilon}, O(1))$

Bounds for Sorting with $t \geqslant 2$ EM-Tapes

Proposition:

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible.
Then there is an r with $r(N) \in O(\log \frac{N}{s(N)})$ such that $\textsc{Sort} \in \mathrm{ST}(r(N), s(N), 2)$

Proof: Refine Chen and Yap's implementation of Merge-Sort

Main Theorem: *(Grohe, Hernich, S., 2006)*

Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(N) \in o(N)$.
For every r with $r(N) \in o(\log \frac{N}{s(N)})$ we have $\textsc{Sort} \notin \mathrm{ST}(r(N), s(N), O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(r(N), s(N), O(1))$

Corollary:

(a) $\textsc{Sort} \notin \mathrm{ST}(o(\log\log N), O(\frac{N}{\log N}), O(1))$; $\textsc{Sort} \in \mathrm{ST}(O(\log\log N), O(\frac{N}{\log N}), O(1))$

(b) *For every ε with* $0 < \varepsilon < 1$ *we have* $\textsc{Sort} \notin \mathrm{ST}(o(\log N), N^{1-\varepsilon}, O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(o(\log N), N^{1-\varepsilon}, O(1))$

Bounds for Sorting with $t \geqslant 2$ EM-Tapes

Proposition:

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible.
Then there is an r with $r(N) \in O(\log \frac{N}{s(N)})$ such that $\textsc{Sort} \in \mathrm{ST}(r(N), s(N), 2)$

Proof: Refine Chen and Yap's implementation of Merge-Sort

Main Theorem: *(Grohe, Hernich, S., 2006)*

Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(N) \in o(N)$.
For every r with $r(N) \in o(\log \frac{N}{s(N)})$ we have $\textsc{Sort} \notin \mathrm{ST}(r(N), s(N), O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(r(N), s(N), O(1))$

Corollary:

(a) $\textsc{Sort} \notin \mathrm{ST}(o(\log\log N), O(\frac{N}{\log N}), O(1))$; $\textsc{Sort} \in \mathrm{ST}(O(\log\log N), O(\frac{N}{\log N}), O(1))$

(b) *For every ε with $0 < \varepsilon < 1$ we have $\textsc{Sort} \notin \mathrm{ST}(o(\log N), N^{1-\varepsilon}, O(1))$*
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(o(\log N), N^{1-\varepsilon}, O(1))$

Bounds for Sorting with $t \geqslant 2$ EM-Tapes

Proposition:

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible.
Then there is an r with $r(N) \in O(\log \frac{N}{s(N)})$ such that $\textsc{Sort} \in \mathrm{ST}(r(N), s(N), 2)$

Proof: Refine Chen and Yap's implementation of Merge-Sort

Main Theorem: *(Grohe, Hernich, S., 2006)*

Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(N) \in o(N)$.
For every r with $r(N) \in o(\log \frac{N}{s(N)})$ we have $\textsc{Sort} \notin \mathrm{ST}(r(N), s(N), O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(r(N), s(N), O(1))$

Corollary:

(a) $\textsc{Sort} \notin \mathrm{ST}(o(\log\log N), O(\frac{N}{\log N}), O(1))$; $\textsc{Sort} \in \mathrm{ST}(O(\log\log N), O(\frac{N}{\log N}), O(1))$

(b) *For every ε with* $0 < \varepsilon < 1$ *we have* $\textsc{Sort} \notin \mathrm{ST}(o(\log N), N^{1-\varepsilon}, O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(o(\log N), N^{1-\varepsilon}, O(1))$

Bounds for Sorting with $t \geqslant 2$ EM-Tapes

Proposition:

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible.
Then there is an r with $r(N) \in O(\log \frac{N}{s(N)})$ such that $\textsc{Sort} \in \mathrm{ST}(r(N), s(N), 2)$

Proof: Refine Chen and Yap's implementation of Merge-Sort

Main Theorem: *(Grohe, Hernich, S., 2006)*

Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(N) \in o(N)$.
For every r with $r(N) \in o(\log \frac{N}{s(N)})$ we have $\textsc{Sort} \notin \mathrm{ST}(r(N), s(N), O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(r(N), s(N), O(1))$

Corollary:

(a) $\textsc{Sort} \notin \mathrm{ST}(o(\log\log N), O(\frac{N}{\log N}), O(1))$; $\textsc{Sort} \in \mathrm{ST}(O(\log\log N), O(\frac{N}{\log N}), O(1))$

(b) *For every ε with* $0 < \varepsilon < 1$ *we have* $\textsc{Sort} \notin \mathrm{ST}(o(\log N), N^{1-\varepsilon}, O(1))$
and $\textsc{Multiset-Equality} \notin \mathrm{RST}(o(\log N), N^{1-\varepsilon}, O(1))$

Main Lower Bound Theorem: Proof Ideas

Theorem: If $s(N) \in o(N)$ and $r(N) \in o(\log \frac{N}{s(N)})$, then

MULTISET-EQUALITY $\notin \mathrm{RST}\big(r(N), s(N), O(1)\big)$

Proof ideas:

1. New machine model: List Machines
   - can only compare and move around input strings
     ($\rightsquigarrow$ weaker than TMs)
   - non-uniform & lots of states and tape symbols
     ($\rightsquigarrow$ stronger than TMs)
2. Simulate (r, s, t)-bounded TMs by list machines.
3. Prove that list machines cannot solve MULTISET-EQUALITY
   (. . . use combinatorics).

List Machines (1/2)

List machines are similar to Turing machines,
with the following important differences:

- non-uniform:
  The input consists of m Bitstrings, each of length n, for fixed m, n.

- Lists instead of tapes. In particular, a new cell can be inserted between two existing cells.

- Each list cell contains strings over the alphabet

$$\mathbb{A} \;=\; I \cup \text{states} \cup \{\langle , \rangle\} \cup C\,,$$

  where $I = \{0,1\}^n$ is the set of potential input strings and
  C is a finite set of "nondeterministic choices".

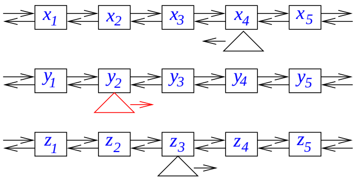
List Machines (2/2)

- The transition function only determines the list machine's new state and the head movements; and not what is written into the list cells.
- If (at least) one head moves, a new list cell is inserted behind the current head position on (almost) every list. This list cell contains the current state and the contents of all list cells that are currently being read.

List Machines (2/2)

- The transition function only determines the list machine's new state and the head movements; and not what is written into the list cells.
- If (at least) one head moves, a new list cell is inserted behind the current head position on (almost) every list. This list cell contains the current state and the contents of all list cells that are currently being read.

Example: $\delta(q, x_4, y_2, z_3, c) = (q', \downarrow, \rightarrow, \downarrow)$

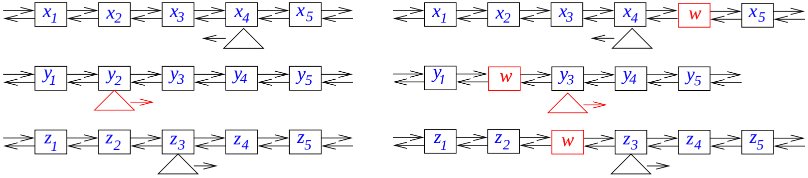


current state: q

List Machines (2/2)

- The transition function only determines the list machine's new state and the head movements; and not what is written into the list cells.
- If (at least) one head moves, a new list cell is inserted behind the current head position on (almost) every list. This list cell contains the current state and the contents of all list cells that are currently being read.

Example: $\delta(q, x_4, y_2, z_3, c) = (q', \downarrow, \rightarrow, \downarrow)$



current state: q $\qquad$ $w := q \langle x_4 \rangle \langle y_2 \rangle \langle z_3 \rangle \langle c \rangle$

The Simulation Lemma (TM $\rightsquigarrow$ LM)

Lemma:

Every (r, s, t)-bounded TM can be simulated by a family of LMs, i.e., for every $m, n \in \mathbb{N}$ there is a LM $L_{m,n}$ which

- *for all inputs $(x_1, \ldots, x_m)$ with $x_i \in \{0,1\}^n$, $L_{m,n}$ has the same acceptance probability as the TM with input $x_1\# \cdots x_m\#$,*
- *has t lists,*
- *has the same number of head reversals as the TM (i.e., $r(N)$ for $N := m \cdot (n+1)$),*
- *has at most $2^{O(r(N) \cdot s(N) + \log N)}$ states.*

Proof Sketch (Simulation Lemma)

- One list for each external memory tape.
- Each list cell represents an entire block of the corresponding TM tape.

  Problem: Block boundaries change throughout the simulation.

- Each state of the LM represents
  - current state q of the TM
  - contents and head positions of the short tapes

    string of length $O(s(N))$
  - head positions and block boundaries of the long tapes

    length of long tapes $\leqslant N \cdot 2^{O(r(N)\cdot s(N))}$

$\Longrightarrow$ in total, $2^{O(r(N)\cdot s(N)+\log N)}$ LM-states will suffice.

Proof Sketch (Simulation Lemma)

- One list for each external memory tape.
- Each list cell represents an entire block of the corresponding TM tape.

  Problem: Block boundaries change throughout the simulation.

- Each state of the LM represents
  - current state q of the TM
  - contents and head positions of the short tapes

    string of length $O(s(N))$
  - head positions and block boundaries of the long tapes

    length of long tapes $\leqslant N \cdot 2^{O(r(N) \cdot s(N))}$

$\Longrightarrow$ in total, $2^{O(r(N) \cdot s(N) + \log N)}$ LM-states will suffice.

Proof Sketch (Simulation Lemma)

- One list for each external memory tape.
- Each list cell represents an entire block of the corresponding TM tape.

  Problem: Block boundaries change throughout the simulation.

- Each state of the LM represents
  - current state q of the TM
  - contents and head positions of the short tapes

    string of length $O(s(N))$
  - head positions and block boundaries of the long tapes

    length of long tapes $\leqslant N \cdot 2^{O(r(N) \cdot s(N))}$

$\Longrightarrow$ in total, $2^{O(r(N) \cdot s(N) + \log N)}$ LM-states will suffice.

Proof Sketch (Simulation Lemma)

- One list for each external memory tape.
- Each list cell represents an entire block of the corresponding TM tape.

  Problem: Block boundaries change throughout the simulation.

- Each state of the LM represents
  - current state q of the TM
  - contents and head positions of the short tapes

    string of length $O(s(N))$
  - head positions and block boundaries of the long tapes

    length of long tapes $\leqslant N \cdot 2^{O(r(N) \cdot s(N))}$

$\Longrightarrow$ in total, $2^{O(r(N) \cdot s(N) + \log N)}$ LM-states will suffice.

Proof Sketch (Simulation Lemma)

- One list for each external memory tape.
- Each list cell represents an entire block of the corresponding TM tape.

  Problem: Block boundaries change throughout the simulation.
- Each state of the LM represents
  - current state q of the TM
  - contents and head positions of the short tapes

    string of length $O(s(N))$
  - head positions and block boundaries of the long tapes

    length of long tapes $\leqslant N \cdot 2^{O(r(N) \cdot s(N))}$

$\Longrightarrow$ in total, $2^{O(r(N) \cdot s(N) + \log N)}$ LM-states will suffice.

Lower Bound for Sorting with List Machines

Lemma:

Let k, m, n, r, t be such that

$$t \geqslant 2,\ m \geqslant 24{\cdot}(t{+}1)^{4^r} + 1,\ k \geqslant 2m{+}3,\ n \geqslant 1 + (m^2 + 1) \cdot \log(2k).$$

Then there is no r-bounded LM with t lists and $\leqslant k$ states that solves the MULTISET-EQUALITY *problem for $2m$ inputs from $\{0,1\}^n$.*

Proof idea:

- Skeleton of a computation:
  replace stings (size n) by their indices (size $\log m$)
- The skeleton determines the flow of information during a computation of a list machine.
- Use counting arguments to show that there are distinct input sequences that have the same skeleton, in which certain strings should be compared, but aren't.

Lower Bound for Sorting with List Machines

Lemma:

Let k, m, n, r, t be such that

$$t \geqslant 2,\ m \geqslant 24{\cdot}(t{+}1)^{4r} + 1,\ k \geqslant 2m{+}3,\ n \geqslant 1 + (m^2 + 1) \cdot \log(2k).$$

Then there is no r-bounded LM with t lists and $\leqslant k$ states that solves the MULTISET-EQUALITY *problem for $2m$ inputs from $\{0,1\}^n$.*

Proof idea:

- Skeleton of a computation:
  replace stings (size n) by their indices (size $\log m$)
- The skeleton determines the flow of information during a computation of a list machine.
- Use counting arguments to show that there are distinct input sequences that have the same skeleton, in which certain strings should be compared, but aren't.

Lower Bound for Sorting with Turing Machines

Simulation TM $\rightsquigarrow$ LM

$+$

Lower bound for MULTISET-EQUALITY with list machines

$$\Downarrow$$

MULTISET-EQUALITY $\notin$ $\mathrm{RST}\Big(r(N), s(N), O(1)\Big)$

if $s(N) \in o(N)$ and $r(N) \in o(\log \frac{N}{s(N)})$

Overview

A model based on Turing machines
- The ST-model
- One external memory tape
- Several external memory tapes
- Future Tasks

A model for database query processing: Finite Cursor Machines

Future Tasks

- Show lower bounds for randomized computations with two-sided bounded error.

- Show lower bounds for appropriate problems in a setting where $\Omega(\log N)$ head reversals and several EM-tapes are available.

  Caveat: It is known that $\textsc{Logspace} \subseteq \mathrm{ST}(O(\log N), O(1), 2)$.

Overview

A model based on Turing machines

- The ST-model
- One external memory tape
- Several external memory tapes
- Future Tasks

A model for database query processing: Finite Cursor Machines

Finite Cursor Machines

ICDT'07 — Joint work with
Martin Grohe, Yuri Gurevich, Dirk Leinders, Jerzy Tyszkiewicz, Jan Van den Bussche

- an abstract model for database query processing
- based on Abstract State Machines (instead of Turing machines)
- Fixed:
  - a background structure $\mathcal{U}$ that consists of an infinite set U of potential database entries, and some functions and predicates on U (e.g., $\mathcal{U} = (\mathbb{N}, <, +, \times)$)
  - a database schema σ that consists of a finite number of relation symbols $R_1, \ldots, R_t$ (of arities $r_1, \ldots, r_t$)
- Input of a FCM: a database D of schema σ
  - D is a collection of t tables $R_1^D, \ldots R_t^D$ (for a fixed $t \in \mathbb{N}$), where each table R_i^D is a list of elements from U^{r_i}
  - $n :=$ the "length" of the input, i.e., the total number of tuples in D
- on every input table, the FCM has a fixed number of cursors which can only move in one direction: from top to bottom
- apart from this, the FCM also has an internal memory consisting of a constant number of "modes" (comparable to a TM's states) and of a register for storing up to $o(n)$ many bits.

Finite Cursor Machines

ICDT'07 — Joint work with
Martin Grohe, Yuri Gurevich, Dirk Leinders, Jerzy Tyszkiewicz, Jan Van den Bussche

- an abstract model for database query processing
- based on Abstract State Machines (instead of Turing machines)
- Fixed:
  - a background structure $\mathcal{U}$ that consists of an infinite set U of potential database entries, and some functions and predicates on U (e.g., $\mathcal{U} = (\mathbb{N}, <, +, \times)$)
  - a database schema σ that consists of a finite number of relation symbols $R_1, \ldots, R_t$ (of arities $r_1, \ldots, r_t$)
- Input of a FCM: a database D of schema σ
  - D is a collection of t tables $R_1^D, \ldots R_t^D$ (for a fixed $t \in \mathbb{N}$), where each table R_i^D is a list of elements from U^{r_i}
  - $n :=$ the "length" of the input, i.e., the total number of tuples in D
- on every input table, the FCM has a fixed number of cursors which can only move in one direction: from top to bottom
- apart from this, the FCM also has an internal memory consisting of a constant number of "modes" (comparable to a TM's states) and of a register for storing up to $o(n)$ many bits.

Finite Cursor Machines

ICDT'07 — Joint work with
Martin Grohe, Yuri Gurevich, Dirk Leinders, Jerzy Tyszkiewicz, Jan Van den Bussche

- an abstract model for database query processing
- based on Abstract State Machines (instead of Turing machines)
- Fixed:
  - a background structure $\mathcal{U}$ that consists of an infinite set U of potential database entries, and some functions and predicates on U (e.g., $\mathcal{U} = (\mathbb{N}, <, +, \times)$)
  - a database schema σ that consists of a finite number of relation symbols $R_1, \ldots, R_t$ (of arities $r_1, \ldots, r_t$)
- Input of a FCM: a database D of schema σ
  - D is a collection of t tables $R_1^D, \ldots R_t^D$ (for a fixed $t \in \mathbb{N}$), where each table R_i^D is a list of elements from U^{r_i}
  - $n :=$ the "length" of the input, i.e., the total number of tuples in D
- on every input table, the FCM has a fixed number of cursors which can only move in one direction: from top to bottom
- apart from this, the FCM also has an internal memory consisting of a constant number of "modes" (comparable to a TM's states) and of a register for storing up to $o(n)$ many bits.

Finite Cursor Machines

ICDT'07 — Joint work with
Martin Grohe, Yuri Gurevich, Dirk Leinders, Jerzy Tyszkiewicz, Jan Van den Bussche

- an abstract model for database query processing
- based on Abstract State Machines (instead of Turing machines)
- Fixed:
  - a background structure $\mathcal{U}$ that consists of an infinite set U of potential database entries, and some functions and predicates on U (e.g., $\mathcal{U} = (\mathbb{N}, <, +, \times)$)
  - a database schema σ that consists of a finite number of relation symbols $R_1, \dots, R_t$ (of arities $r_1, \dots, r_t$)
- Input of a FCM: a database D of schema σ
  - D is a collection of t tables $R_1^D, \dots R_t^D$ (for a fixed $t \in \mathbb{N}$), where each table R_i^D is a list of elements from U^{r_i}
  - $n :=$ the "length" of the input, i.e., the total number of tuples in D
- on every input table, the FCM has a fixed number of cursors which can only move in one direction: from top to bottom
- apart from this, the FCM also has an internal memory consisting of a constant number of "modes" (comparable to a TM's states) and of a register for storing up to $o(n)$ many bits.

Easy Observations

Consider the operators from Relational Algebra

- Selection $\sigma_{i=j}(R)$ can be implemented by a FCM
- Union $R_1 \cup R_2$ and Projection $\pi_J(R)$ can be implemented by a FCM, provided that input tables are ordered
- Joins are NOT computable by FCMs, because the output size of a join can be quadratic, and FCMs can output only a linear number of different tuples
- Window Joins for a fixed window size w can be computed by an FCM
- Semijoins $R \ltimes_\theta S$ can be computed by an FCM, provided that input tables are ordered $R \ltimes_\theta S := \{t \in R : \text{there is an } s \in S \text{ such that } \theta(t, s)\}$

> *Corollary:*
>
> *Each Semijoin Algebra query can be computed by a composition of FCMs and sorting operations.*

Question: Are intermediate sorting steps really necessary?

Easy Observations

Consider the operators from Relational Algebra

- Selection $\sigma_{i=j}(R)$ can be implemented by a FCM
- Union $R_1 \cup R_2$ and Projection $\pi_J(R)$ can be implemented by a FCM, provided that input tables are ordered
- Joins are NOT computable by FCMs, because the output size of a join can be quadratic, and FCMs can output only a linear number of different tuples
- Window Joins for a fixed window size w can be computed by an FCM
- Semijoins $R \ltimes_\theta S$ can be computed by an FCM, provided that input tables are ordered $R \ltimes_\theta S := \{t \in R : \text{there is an } s \in S \text{ such that } \theta(t, s)\}$

> *Corollary:*
>
> *Each Semijoin Algebra query can be computed by a composition of FCMs and sorting operations.*

Question: Are intermediate sorting steps really necessary?

Main Result

Question: Are intermediate sorting steps really necessary?

Answer: Yes . . .

Theorem: *(Grohe, Gurevich, Leinders, S., Tyszkiewicz, Van den Bussche, ICDT'07)*

The query

$$Is\ \ R \ltimes_{x_1=y_1} (S \ltimes_{x_2=y_1} T)\ \ nonempty?$$

where R and T are unary and S in binary, is not computable by an FCM (even if the FCM is allowed to have as input all sorted versions of the input relations).

Open Question

Is there a Boolean query from Relational Algebra (or, equivalently, a sentence of first-order logic), that cannot be computed by any composition of FCMs and sorting operations?

Conjecture: Yes.