A Constant-Space Model of Computation for First-Order Queries

Steven Lindell
Haverford College
USA

Problems

A problem (in Computer Science) specifies an input/output relationship, \( P \subseteq I \times O \).

(input) \( P \)  (output)

How are input/output represented as data?

Multiplication

Example:
(not first-order)
ordering matters
data are strings
MSB > LSB
\[ \begin{array}{c}
\times \\
\text{ordered bits}
\end{array} \]
\[ d_{n-1} \ldots d_0 \times e_{n-1} \ldots e_0 \]
\[ \begin{array}{c}
\text{MSB} \\
\text{LSB}
\end{array} \]

Transitive Closure

Example:
(not first-order)
order invariant
data are relations
\( E \subseteq V \times V \)
\( E^* \subseteq V \times V \)

unordered graphs
matrix operation
(independent of simultaneous row/column permutations)

Complexity: Machine Computability

input data (finite string) \( \xrightarrow{M} \) output data (finite string)

\( M \) is \{time/space\} resource-bounded.
[semantic restriction]

Complexity: Logical expressibility

input data (finite structure) \( \xrightarrow{\phi} \) output data (finite relation)

\( \phi \) is \{SO, FP, FO\} construct-bounded
[syntactic restriction]
Logical Definability

second-order = polynomial-hierarchy
fixed-point(<) = polynomial-time = P
first-order(+, *) = constant (parallel) time \(\subseteq\) L

Logical Framework

Finite relational structure:
\(\langle A, R_1, \ldots, R_k, Q \rangle\)
finite domain
relations
query (defines a new relation)

First-order queries:
\(x, y, z, \ldots\) individual variables (over \(A\))
\(=, R_i\) atomics
\(\land, \lor, \lnot\) boolean connectives
\(\exists x, \forall x\) quantification (over \(A\))

Example: Graph Simplicity

directed graph: \(G = \langle V, E \rangle\)
(binary relation) \(E \subseteq V \times V\)

Simplicity:
\(-\exists x \in V [E(x, x)] \land \) no self-loops
\(\forall y \forall z [E(y, z) \rightarrow E(z, y)] \) undirected edges

Example: Linear Ordering

total linear order:

\(- (x < x) \) irreflexive
\((x \neq y) \rightarrow [(x < y) \lor (y < x)] \) total
\([(x < y) \land (y < z)] \rightarrow (x < z) \) transitive

Binary String Structures

\(\langle \{0, 1, \ldots, n - 1\}, <, U \rangle\)
set of positions
orders the positions
indicates the locations of 0's and 1's

Example: \(w = 1010\) is represented by
\(\langle \{0, 1, 2, 3\}, <, \{0, 2\} \rangle\), with \(0 < 1 < 2 < 3\).

Arithmetic & bit

Definition: For \(w \in \{0, 1\}^*\),
\(\langle |w|, <, \text{bit}, \{i : w_i = 1\} \rangle\) where
\(\text{bit}(i, j) \Leftrightarrow \text{the } j\text{th bit of } i \text{ is 1} \) (numerical predicate)

\(0 < 1 < \ldots < (n - 1)\)
\(\langle n, +, \ast \rangle\)
\(0 \leq a, b, c < n\)
\(a + b = c\)
\(a \ast b = c\)
Examples

$n$ is even? $\in$ FO$(+, \neg) \setminus$ FO$(\neg)$

$n$ is prime? $\in$ FO$(+, *, \neg)$ \setminus$ FO$(+)$

$a \land b = c \in$ FO$(+, *, \neg)$

FO$(\neg, \text{bit}) = FO(+, *)$

\text{PARITY} \not\in$ FO(any numerical predicates, $U$) \cite{FSS}
(asks if $|U|$ is even)

---

Logspace Computation

Off-line Turing machine (sub-linear space):

- size = $n$
- read-only input tape
- heads
- read/write work tape
- space = $O(\log n)$
- space = $O(1)$

---

Bidirectional multi-head DFA

- read-only input tape
- finite state control

Each head can be thought of as an integer “cursor” holding $\log n$ bits of storage, so this model is equivalent to logspace.

---

Constant (parallel) time

AC$^0$

Circuit model (non-uniform):

constant depth polynomial size

inputs w/ negations

---

PRAM model (uniform)

resolve conflicts in various ways

- global read/write memory
- $O(n^{\Omega(1)})$ processors

FO$(+, *)$ = PRAM($O(1)$ time) \cite{Immerman}

---

Regular Languages

On word models

\text{REG} = \text{mSO}(\neg)$

star-free \text{REG} = \text{FO}(\neg)$
Inclusions

EQUAL = \{ w: \text{equal # of 0's and 1's} \} \in ALOGTIME\...

ALOGTIME
(uniform NC^1)

MIDDLE = \{ 0^{n1}\cdot n = 0, 1, \ldots \} \in FO(<, \text{bit}) \cap REG

PARITY = \{ w: \text{even # of 1's} \} \in \text{REG} \cap FO(<, \text{bit})

EVEN = \{ w: |w| \text{ is even} \} \in FO(<, \text{mod})

Finite-State Machines

FSM:
single oblivious head
flip/flops & gates

Chart
finite automata
proposed model

I/O access
single scan
multiple passes

control
state machine
restricted mechanism

Proposed Model

multiple oblivious heads
destructive read

Constant-Space Machine (hardware)

read-only input tape
... (one bit)
flip-flop

finite control

finite control

Constant-Space Program (software)

b
boolean variable
I[h]
input bit under head h
i < j
head i is left of head j
\text{bit}(i,j)
jth bit of head i's counter
\text{AND, OR, NOT}
to form expressions
b := e
boolean assignment
P; Q;
sequential composition
LOOP h
sequential iteration
P;
(head h scans whole tape)
I\text{F} \ e \ \text{THEN} \ P;
conditional (not allowed)

• Program syntax guarantees: obliviousness; read-once
• Finiteness implies: constant-space; polynomial-length

Sample Program: Parity

Compute the \textit{parity} of \( a = a_1 \ldots a_n \):\

\[
p := 0; \quad \{ \text{initialize} \}
\]

\[
\text{LOOP } h \quad \{ \text{LSB to MSB} \}
\]

\[
p := a(h) \text{ XOR } p; \quad \{ \text{exclusive-or} \}
\]

Note: \( p \) violates the read-once condition in \( \oplus \).
Sample Program: Binary addition

Binary addition: \(a_n \ldots a_1 + b_n \ldots b_1 = c_n \ldots c_1\)

\[
c := 0; \quad \text{(initialize carry)}
\]

LOOP \(h\) \{from LSB to MSB\}

\[
s(h) := a(h) \text{ XOR } b(h) \text{ XOR } c
\]

\[
c := a(h) \text{ AND } b(h) \text{ OR } c \text{ AND } ((a(h) \text{ OR } b(h))
\]

- \(c\) in output statement doesn’t count since \(s\) is write-only
- \(c\) occurs once in re-parenthesized majority function

Main Result

Theorem:

A (binary) language \(L \subseteq \{0, 1\}^*\) is recognizable by a read-once constant-space program if and only if it is definable by a first-order formula with arithmetic

Time / Space Duality

Corollary (Duality principle): A problem is computable by a read-once constant-space serial algorithm if and only if it is computable by a constant-time parallel algorithm [extends previously known duality to below \(O(\log n)\)].

Carry look-ahead (binary addition)

\[
s(i) = a(i) \oplus b(i) \oplus c(i) \quad \text{where}
\]

\[
c(i) = (\exists j)[j < i \land a(j) \land \exists b(j) \land
\]

\[
(\forall k)[j < k < i \rightarrow (a(k) \lor b(k))]\]

Extensions

Constant-depth threshold circuits:

- Output on iff more than \(k\) inputs are on
- \(k = 0\) OR gate
- \(k = n-1\) AND gate

Variable threshold logic (uniform \(TC^0\)):

\[
(\exists^i x) \theta(x) \quad 0 \leq i \leq n-1
\]

Threshold quantifier

Constant Result

Question

Is there a uniform, deterministic, sequential model of computation for \(VT(<,\ast)\)? What is the dual to the serial algorithm for multiplication? (see Slide 1 for picture)

\[
s := 0 \quad \text{initialize partial sum}
\]

LOOP \(i\) FROM 0 TO \(2^n-1\) \{LSB to MSB of answer\}

\[
s := a(i)^* e(i-j) \quad \text{do sum for } i\text{th column}
\]

\[
\rho(i) := s \mod 2 \quad \text{add next term}
\]

\[
s := s \div 2 \quad \text{product bit}
\]

\[
s := s \div 2 \quad \text{carry to next column}