Linear-time algorithms
for Monadic Logic
Steve Lindell  slindell@haverford.edu
Haverford College, Department of Computer Science

Descriptive complexity studies the asymptotic computational effort required to evaluate logical queries on finite databases, with a focus on queries expressed by first-order or fixed-point formulas. In general, these queries require a polynomial amount of time with respect to the size of the structure. We illustrate the special case of how first-order sentences can be evaluated in linear-time on ordinary data structures. Extending this to monadic fixed-point formulas leads to the possibility of providing a long sought after logical characterization of linear-time computability.

Acknowledgements
• My colleague Scott Weinstein for many discussions
• My wife Suzanne for reviewing and proofreading

First-order queries

Vocabulary
Variables: \( x, y, z \) range over nodes
Constants: \( c, d \) are fixed nodes
Properties: \( P(x), Q(x) \) to query content bits
Functions: \( F(x), G(x) \) to link nodes by pointers

Combine in usual way to form terms by functional composition and predicates by combinations of Boolean connectives applied to properties of terms. Then add existential/universal quantification over the domain.

Example: The induced graph is a union of cycles: \( \forall x \langle G(F(x)) = x \rangle \) i.e. \( G = F^{-1} \)

Computational complexity

<table>
<thead>
<tr>
<th>(data) structure</th>
<th>binary string</th>
<th>machine</th>
<th>(boolean) query</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D )</td>
<td>( B )</td>
<td>( M )</td>
<td>( D \models \Phi )</td>
</tr>
</tbody>
</table>

Important: For \( M \) to compute \( \Phi \) it must be isomorphism invariant.
Typically, \( \Phi \) is expressed in a fixed-point extension of first-order logic.

Descriptive complexity: systematic study of connections between definability and complexity on (WLOG for us) ordered structures.

\( P = \) polynomial-time = LFP (least fixed-point logic) [Immerman]
\( L \) = logarithmic-space = DTC (deterministic transitive-closure logic)

Where does that leave first-order logic? With arithmetic, \( \text{FO}(+, \times) = O(1) \) time on parallel \( \text{RAM} = O(n) \) space on read-only sequential \( \text{RAM} \)
Still requires log \( n \) space and \( n^{O(1)} \) work. Can we do better? Yes!

Logical definability

In a common vocabulary \( V \):
Fix a particular logical query \( \phi(x) \)

The result defines on \( D \) either a proposition predicate a marking of nodes

Vary a general data structure \( D \)
input \( D \models \phi(x) \) output \( \text{true} \) \( \text{false} \)

Question: How long does it take to evaluate \( \phi \) with respect to the size of \( D \)?
Answer: Depends on the syntactic power of the logic and the shape of the structure.

(Historical Example) Monadic second-order logic on strings (or trees) is \( O(n) \).
E.g. binary parity: \( \exists! Q(x) \land \forall x \leq t, P(x) \iff [Q(x) \iff Q(S(x)) \land [P(t) \iff Q(t)] \)

Seese’s Theorem

Compute first-order sentences in \textit{linear-time} on bounded-degree graphs.

Basic idea: Fix the degree bound \( d \) (in our case it’s determined by our representation of nodes in a doubly-linked data structure) and the quantifier depth \( q \) of the sentence \( \Phi \). Then \( \Phi \) is equivalent to a Boolean combination of sentences which say:

\[ \exists x \text{ says there are at least threshold } t \text{ is which satisfy…} \]
\[ \tau_s(x) \text{ is a local (quantifier-free) formula which depends only on the neighborhood of radius } r \text{ about } x. \]

Linear-time algorithm: For each node \( x \), go out to the radius \( r = 2^q \) (constant time because the neighborhood is bounded in size), and determine if \( \tau_s(x) \) is true. Count only up to the threshold \( t = qd^q + 1 \).
**Observation:** On graphs of bounded-degree, there are only a fixed number $t_1 \ldots t_n$ of non-isomorphic neighborhoods of radius $r$.

**Define:** The global $(r, t)$ type of a structure is composed of the numbers of nodes of each kind of radius $r$ neighborhood up to the threshold $t$.

<table>
<thead>
<tr>
<th>$(r, t)$</th>
<th>1</th>
<th>2</th>
<th>\ldots</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>*</td>
<td>*</td>
<td>\ldots</td>
<td>*</td>
</tr>
<tr>
<td>$t_2$</td>
<td>*</td>
<td>*</td>
<td>\ldots</td>
<td>*</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\ldots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$t_n$</td>
<td>*</td>
<td>*</td>
<td>\ldots</td>
<td>*</td>
</tr>
</tbody>
</table>

The table always overflows for sufficiently large structures because it contains only a finite number of dots.

**Lemma (for sentences):** Two structures with the same global type must agree on $\phi$. It suffices to recognize which global types satisfy $\phi$.

**Extension (for formulas):** Whether an element $x$ satisfies $\phi(x)$ depends only on the global $(r', t')$ type together with its local $r'$-type.

---

**Monotonicity Lemma:**

- **Positivity implies $\phi$ is monotone:** $S \subseteq S' \Rightarrow \phi(S) \subseteq \phi(S')$
- **Naive algorithm for $\phi^*$ is $O(n^2)$:** $S \leftarrow \emptyset$ $S \leftarrow \phi(S)$

Stages $\emptyset \subseteq \phi^1 \subseteq \phi^2 \subseteq \ldots \subseteq \phi^n$ increase while staying below fixed-point.

**New Idea:** cannot go stage by stage. How? Answer: Any monotone method using $\phi$.

**Lemma:** If $S \subseteq \phi^r$ is strictly below the fixed-point, then:

(a) $\exists v \in \phi(S) \bullet v \notin S$ (there are new things); and
(b) $\forall v \in \phi(S) \bullet v \notin \phi^r$ (every thing is safe).

**Proof:** As a direct consequence of monotonicity, $S \subseteq \phi^r$ implies $\phi(S) \subseteq \phi(\phi^r)$, which is a restatement of (b). The fact that $S \neq \phi^r$ implies there is a largest $r$ such that $\phi^r \subseteq S$, which means $\phi^{r+1}$ contains things that are not in $S$. Since $\phi^{r+1} \subseteq \phi(\phi^r)$ by monotonicity, this implies that $\phi^r \cup \{x\} \subseteq \phi(S)$, which shows that (a). QED

In other words, starting from $\emptyset$ and increasing in the $\phi$ direction will always reach the fixed-point. In particular, we can add just one element at each step.

---

**Linear time algorithm:**

Fix an $S$-positive first-order formula $\phi(x; S)$ of quantifier-depth $q$.

- **Goal:** given a graph $G$ of degree $d$, mark all nodes $v \ni G \models \phi^*(v)$.

1. **Step 0 (Initialize):** Scan the radius $r$ neighborhood of each vertex $v$ to determine its $r$-type $\tau(v)$, and insert it into $L_{0\tau}$: Time: $O(n)$ once.
2. **Step 1 (Determine markable types):** From the threshold $t$ numbers $\min(|L_{0\tau}|, t)$ in our finite control, determine the set $M = \{\tau(v) : G \models \psi(v, S)\}$ of types whose elements would satisfy $\phi(S)$: Time: $O(1)$.
3. **Step 2 (Choose node to mark):** From among the unmarked types $\tau$ in $M$ pick the head element $v_0$ off any non-empty list $L_{\tau}$. If all such lists are empty, we are done. Time: $O(1)$.
4. **Step 3 (Update the types):** Since distance is symmetric, marking $v_0$ changes only the types of the fixed number of nodes within radius $r$ of $v_0$. Remove each such $v$ from its old list $L_{\tau(v)}$ and put it into its new list $L'_{\tau(v)}$. Goto Step 1: Time: $O(1)$.

---

**Summary of Results:**

On bounded-degree graphs, we have extended the containment of first-order queries in linear-time to monadic fixed-point. I.e.: $FO \subseteq O(n)$ is improved to monadic FP $\subseteq O(n)$

New technique: combines locality of first-order formulas and monotonicity of inductions in a way that allows sequential evaluation by a graph automata operating directly on the input/output structure.

**Open Questions**

1. **Conjecture:** finite-visit graph automata contained in mFP (easily seen to be true when there is no pointer manipulation)
2. **Conjecture:** bounded-degree FP $\subseteq O(n)$ (very natural: cannot store unbounded degree information anyway)
3. **Question:** Is there a logical characterization of linear-time computation on Kolmogorov machines?