

6/25/2003 1

Linear-time algorithms for Monadic Logic

Steven 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

6/25/2003 2

Data Structures

Finite collections of nodes: Uniformity: fixed width

0	.. contents ..	1
•	.. links ..	•

Assumptions: doubly-linked so that both in-degree and out-degree are bounded (i.e. underlying graph is edge-symmetric). Nil pointers have been omitted.

Examples:

list

tree

grid

6/25/2003 3

First-order queries

Vocabulary

$P(r)$...bits...	$Q(r)$
$F(r)$...pointers...	$G(r)$

Variables: $x, y, z \dots$ range over nodes
Constants: $c, d \dots$ are fixed nodes
Properties: $P(x), Q(x) \dots$ to query content bits
Functions: $F(x), G(x) \dots$ 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: $\forall x [G(F(x)) = x]$ i.e. $G = F^{-1}$

6/25/2003 4

Logical definability

In a common vocabulary V : Fix a particular logical query ϕ . The result defines on D either a proposition or a predicate.

Vary a general data structure D → input $D \models \phi(x)$ → output $\begin{cases} \text{true} \\ \text{false} \end{cases}$ a marking of nodes

Question: How long does it take to evaluate ϕ 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: $\exists Q \rightarrow Q(s) \wedge \forall x \neq t \rightarrow P(x) \leftrightarrow [Q(x) \leftrightarrow Q(S(x))] \wedge [P(t) \leftrightarrow Q(t)]$

0111 is odd **1001 is even**

$$P: \begin{matrix} \overset{s}{1} & \xrightarrow{S} & \overset{s}{0} & \xrightarrow{S} & \overset{s}{1} & \xrightarrow{S} & \overset{t}{0} \\ \oplus & & \oplus & & \oplus & & \oplus \end{matrix}$$

$$Q: \begin{matrix} \boxed{F} = \boxed{T} = \boxed{T} = \boxed{F} = F \end{matrix}$$

6/25/2003 5

Computational complexity

(data structure) D $\xrightarrow{\text{code}}$ (binary string) B $\xrightarrow{\text{input}}$ (machine) M $\xrightarrow{\text{output}}$ (boolean query) $D \models \Phi$

Important: For M to compute Φ it must be *isomorphism invariant*. Typically, Φ 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, $FO(+, \times) = O(1)$ time on parallel RAM = $O(1)$ space on read-once sequential RAM
 Still requires $\log n$ space and $n^{O(1)}$ work. Can we do better? Yes!

6/25/2003 6

Seese's Theorem

Compute first-order sentences in *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 ϕ . Then ϕ is equivalent to a Boolean combination of sentences which say:

$\exists! x$ says there are at least threshold t x 's which satisfy...

$(\exists^t x) \tau_r(x)$

$\tau_r(x)$ is a local (quantifier-free) formula which depends only on the neighborhood of radius r 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 $\tau_r(x)$ is true. Count only up to the threshold $t = qd^{q+1}$.

6/25/2003 7

Hanf's Lemma

Observation: On graphs of bounded-degree, there are only a fixed number $\tau_1 \dots \tau_N$ of non-isomorphic neighborhoods of radius r .

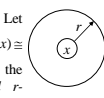
Define: The *global* (r, t) type of a structure is composed of the numbers of nodes of each kind of radius r neighborhood up to the threshold t :

(r, t)	1	2	...	t
τ_1	•			
\dots				
τ	•	•	•	•
\dots				
τ_N	•	•	•	•

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

Therefore, there are only finitely many different global (r, t) types.

Let $\tau(x) \equiv$ be the local r -type of x



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

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

6/25/2003 8

Monadic fixed-point logic

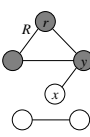
Example: Which nodes are reachable from a constant root via (symmetric) links?

Idea: Mark adjacent nodes by a *new* monadic predicate R , starting from the root r .

Recursion: $R(x) \leftrightarrow \phi(x; R) \equiv x = r \vee \exists y R(y) \wedge y \sim x$
 where $y \sim x$ abbreviates $x = F(y) \vee x = G(y) \dots$

Take the least fixed-point with respect to R : $\phi^c(x)$

Solution: The query $\forall x \phi^c(x)$ expresses connectivity.



Notation: For $S \subseteq D$, let $\phi^D(S) = \{x \in D \mid D \models \phi(x; S)\}$ (we usually drop D).

Definition (Moschovakis): A *monadic elementary induction* is formed by taking a first-order $\phi(x; S)$ positive in S , and using its *stages*:

$$\phi^0 = \emptyset; \phi^{i+1} = \phi(\phi^i)$$

to obtain the *least fixed-point* of $\phi(S) = S$ in at most $n = |D|$ steps:

$$\phi^0 \subseteq \dots \subseteq \phi^i \subseteq \phi^{i+1} \subseteq \dots \subseteq \phi^m = \phi^{m+1} = \phi^\infty$$

6/25/2003 9

Monotonicity Lemma

Positivity implies ϕ is monotone: $S \subseteq S' \Rightarrow \phi(S) \subseteq \phi(S')$

Naïve algorithm for ϕ^∞ is $O(n^2)$: $S \leftarrow \emptyset \quad S \leftarrow \phi(S)$

Stages $\emptyset = \phi^0 \subseteq \dots \subseteq \phi^i \subseteq \phi^{i+1} \subseteq \dots \subseteq \phi^m = \phi^\infty$ increase while staying below fixed-point.

New Idea: *don't go stage by stage.* How? **Answer:** Any monotone method using ϕ .

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

- $\exists s \in \phi(S) \bullet s \notin S$ (ϕ finds something new); and
- $\forall s \in \phi(S) \bullet s \in \phi^\infty$ (everything ϕ finds is safe).

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

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

6/25/2003 10

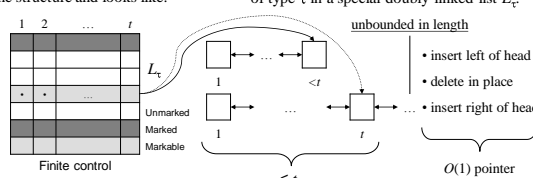
List Automaton

To each node add: a new property field S for marking the fixed-point; and two new link fields.

Automata head walks around the structure and looks like:

For each τ , dynamically maintain all nodes of type τ in a special doubly-linked list L_τ :

- insert left of head
- delete in place
- insert right of head



Key: Keeps track of $\min\{|L_\tau|, t\}$ and an element of type τ (if one exists). Nodes move between lists if/when they change type (because of changes to the marking S).

6/25/2003 11

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 $v \ni G \models \phi^c(v)$.

Step 0 (Initialize): Scan the radius r neighborhood of each vertex v to determine its r -type $\tau(v)$, and insert it into $L_{\tau(v)}$. *Time:* $O(n)$ once.

Step 1 (Determine markable types): From the threshold t numbers $\min\{|L_\tau|, t\}$ in our finite control, determine the set $M = \{\tau(v) \mid G \models \phi(v, S)\}$ of types whose elements would satisfy $\phi(S)$. *Time:* $O(1)$.

Step 2 (Choose node to mark): From among the unmarked types τ in M pick the head element v_0 off any non-empty list L_τ . If all such lists are empty, we are done. *Time:* $O(1)$.

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:* $O(1)$.

6/25/2003 12

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.:

$$FO \subseteq O(n) \quad \text{is improved to} \quad \text{monadic FP} \subseteq O(n)$$

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

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