Linear-time algorithms for Monadic Logic

Steven Lindell
Haverford College, Department of Computer Science
slindell@haverford.edu

Abstract

In general, the computational effort required solving a problem described by a first-order or fixed-point query (logical formula) requires time polynomial in the size of the database (finite structure). We show how a linear-time evaluation algorithm for first-order logic on bounded-degree data structures can be extended to monadic fixed-point formulas, pointing the way to a logical characterization of linear-time computability.

It is well-known that first-order queries can be evaluated in logarithmic-space on arbitrary finite structures. On bounded-degree graphs, it is surprising that first-order sentences can be evaluated in linear-time [1]. In this short presentation we illustrate how to also do this for monadic fixed-point formulas.

Let $f(x; S)$ be a first-order formula of one free variable in the language of graphs (vertices $V$ with one binary edge relation $E$) with equality, with an additional unary relation $S$, appearing only positively. Write $f(S) = \{ s : G \models f(s; S) \}$ where the finite graph $G = (V, E)$ is understood to have degree at most $d$.

**Def:** The least fixed-point of $f$, denoted $f^\omega$, is the smallest relation $S$ satisfying $f(S) = S$.

It can be computed by any monotone method which stays below the fixed-point.

**Lemma:** If $S \subset f^\omega$ is strictly below the fixed-point, then: (a) $\exists s \in f(S) \bullet s \notin S$; & (b) $\forall s \in f(S) \bullet s \in f^\omega$.

In particular, a greedy method can be used in which only one element is thrown in at each step.

**Corollary:** If $s_1 \in f(\{s_1, \ldots, s_k\})$ is a maximal length $k$ sequence of distinct nodes, then $\{s_1, \ldots, s_k\} = f^\omega$.

Realize our graphs now include a unary relation $S$. Define the $r$-type of a node $s$ as the isomorphism type of the neighborhood of radius $r$ about $s$. Use $G_1 \cong_r G_2$ to denote two graphs that realize the same number of $r$-types with the same multiplicity up to a certain threshold $t$.

**Lemma:** Given $f(s)$, there is a radius $r$ and a threshold $t$ such that if $G_1 \cong_r G_2$ are of degree at most $d$, and $s_1$ and $s_2$ have the same $r$-type, then $G_1 \models f(s_1) \iff G_2 \models f(s_2)$.

So whether $G$ satisfies $f(v)$ depends only the quantity up to $t$ of each $r$-type, together with $r$-type of $v$.

**Corollary:** Let $\tau(v)$ be the $r$-type of $v$ in $G$, and let $\#\tau^G = |\{ v \in G : \tau(v) = \tau \}|$ for any $r$-type $\tau$. Then $T(G) = \{ (\tau, \min\{t, \#\tau\}) : \tau \text{ is an } r\text{-type} \}$, together with $\tau(v)$, determines whether $G \models f(v)$.

The algorithm assumes a unit-cost RAM model with $O(\log n)$-bit word size, in which the edge relation of the degree $d$ graph is represented by a doubly-linked incoming and outgoing pointers at each vertex. Our method starts with $S = \emptyset$, and marks one element at a time until $S = f^\omega$. We always select an element in constant time that currently satisfies $f(S)$, keeping track of $S$ by directly marking nodes in $G$, obtaining

**Theorem:** Over bounded-degree graphs, there is a linear-time algorithm to compute monadic fixed-points.