A normal form for first-order logic over doubly-linked data structures

Steven Lindell
Department of Computer Science
Haverford College
2006

We use singulary vocabularies to analyze first-order definability over doubly-linked data structures. Singulary vocabularies contain only monadic predicate and monadic function symbols. A class of mathematical structures in any vocabulary can be interpreted in a singular vocabulary, while preserving the notions of elementary definability and bounded-degree. Doubly-linked data structures are a special case of bounded-degree finite structures in which there are symmetric labeled connections between elements, corresponding closely with physically feasible models of information storage. They can be associated with logical models involving unary relations and bijective functions in what we call an invertible singulary vocabulary. Over classes of these models, there is a normal form for first-order logic which eliminates all quantification of dependent variables. The paper provides a syntactically based proof using counting quantifiers. It also makes precise the notion of implicit calculability for arbitrary arity first-order formulas. Linear-time evaluation of first-order logic over doubly-linked data structures becomes a direct corollary. Included is a discussion of why these special data structures are appropriate for physically realizable models of information.

Introduction

This research grew out of an old question involving the signatures of first-order structures:

- *What is the simplest vocabulary retaining the full expressive power of first-order logic?*

There are many ways to define simplicity, but we chose "smallest arity" as our yardstick, and wondered if the explicit use of pairing in a structure could somehow be avoided. Part of the motivation came from practical applications, because the basic data structures used in Computer Science rely entirely on nodes which store data and pointers to other nodes. These structures arose as models of data storage subject to the practicalities of physical memory devices. This presents a new question regarding the asymptotic scalability of data structures:

- *What are the physically realizable classes of data structures?*

Though we do not extensively argue a convincing answer for this question, we do take the position that doubly-linked data structures are required. Part of the reason for this is the inherent reversibility of connections in space, which in real memory must comprise actual physical links.

Although using pointers to efficiently represent data has clearly proven its practical value in nearly all areas of computing, describing finite models in a theoretical fashion by unary functional vocabularies seems to have been long neglected until the work of [Grandjean]. Indeed, this clever idea has proven its utility by a line of research which has culminated in robustly capturing deterministic linear time computation on random access machines by defining unary functions via algebraic recursion schemes [Grandjean & Schwentick]. In this paper, we are more

interested in how the logical properties of reversibly-linked structures relate to linear-time.

It is well known that structures over any fixed vocabulary can be elementarily interpreted in the language of graphs – i.e. only a single binary relation (with equality) is required.* But perhaps less well recognized is the fact that this can also be accomplished in a singulary vocabulary: one which admits only monadic predicates and monadic function symbols, with equality.

Distinct from monadic logic, which excludes function symbols, first-order singulary logic is expressive enough to interpret elementary definability over any vocabulary whatsoever. The construction is straightforward yet reveals an important fact – in general there will be an unbounded number of references to an unbounded number of nodes. While this presents no problem for purely mathematical studies, it is not realistic for physically scalable models of information. Structures of unbounded-degree cannot be physically realizable. So we restrict our attention to models of information which have a fixed number of symmetric connections between their datum – otherwise known as doubly-linked data structures. While this does not necessarily ensure physical realizability, it comes much closer. The natural vocabulary for these classes is an invertible singulary vocabulary in which each function has an inverse. Within this realm, first-order definability is much simpler than it appears. Our main result states that it is possible to rewrite all first-order formulas in a normal form that eliminates all quantified dependencies between variables.

Theorem: Any first-order formula $\zeta(x_1, \ldots x_k)$ in an invertible singulary vocabulary is equivalent to a Boolean combination of:

1. atomic formulas: $\alpha(x_1, \ldots x_k)$ &
2. numerically quantified sentences: $\exists^n x\, \beta(x)$ where β is quantifier-free.

Over finite structures, the numerically quantified sentences can be computed in linear-time, and the quantifier-free formulas in constant-time. With the appropriate notion of evaluation for formulas with free variables, we obtain a corollary which states that all such first-order formulas are computable in linear-time.

Ideas behind a semantic collapse for first-order queries on bounded-degree structures were motivated from notions of locality due to [Hanf], as treated in [Libkin, Ch.4] and [Immerman, Ch.6]. Ideas for a syntactic collapse of first-order formulas were inspired by [Gaifman], whose normal form is remarkable for its generality. It applies to all relational structures, but quantified dependencies between variables remain [Libkin, p.60]. Further motivation was obtained from the pioneering result of [Seese], which demonstrated a linear-time algorithm for first-order sentences over bounded-degree graphs [Immerman, p.103] and [Libkin, p.101]. A related result was presented in [Lindell] in a different logical setting, applying Hanf's Lemma to first-order sentences, but it did not apply to formulas or discuss linear-time evaluation. Very similar results to ours were independently attained by [Durand & Grandjean].

The outline of the paper is as follows. In the first section we define singulary logic, its syntax and semantics. In the second section we illustrate how any structure can be interpreted as a singulary structure, via a simple first-order transformation from an arbitrary vocabulary to a singulary vocabulary. In the third section we show that relational structures are of bounded-degree if and only if their images as singulary structures are also. In the fourth section we examine finite domains, and see how doubly-linked data structures are really just invertible singulary models. The fifth section contains our main result, a normal form for first-order formulas in invertible singulary vocabularies. In the sixth section we look at consequences of this result from a complexity theoretic point of view, and show any first-order formula can be evaluated in linear-time, regardless of its arity. In the last section we summarize and derive conclusions for the theory of finite models of information.

---

* A fairly complete treatment of this result appears in my Ph.D. dissertation: "The Logical Complexity of Queries on Unordered Graphs," University of California at Los Angeles, 1987.

§1 Singulary logic

According to the Oxford English Dictionary, the word *singulary* is used exclusively in logic, referring to mathematical operations "involving just one element". The term was introduced by W.V. Quine, who preferred it to the more common usage 'unary' because of Latin derivation [Quine]. It became adopted by Alonzo Church who, in concurring with Quine on this etymological point, went so far as to define singulary vocabularies in which all the function and predicate symbols have only one place [Church]. This constraint formally prohibits tupling, and singulary structures can be visualized as directed graphs of bounded out-degree, where each node is colored by the combination of predicate symbols that make it true, and is the origin of exactly one type of outgoing edge for each function symbol.

From a terminological point of view, it will be helpful to distinguish between 'monadic' and 'unary' when referring to intentional and extensional objects respectively, though we make no such distinction for the adjective 'singulary', using it in both situations.

For a symbolic vocabulary to be singulary, all variables must appear alone as the object of predicate or function application (equality is excluded from this restriction so that one variable can appear on each side of an equals sign). More formally,

Definition: A *singulary vocabulary* consists only of monadic predicates and monadic functions

$L = \{P_1, \ldots, P_m, f_1, \ldots, f_k\}$ where each symbol has precisely one place for a variable.

Singulary vocabularies are interpreted by singulary structures in the obvious way:

Definition: A *singulary structure* S with domain $|S|$ has only unary relations and unary functions:

$S = \langle |S|, P_1, \ldots, P_m, f_1, \ldots, f_k \rangle$ where each $P_i \subseteq |S|$, and each f_j: $|S| \to |S|$.

The number of elements in $|S|$ is the *size* of S. As is customary, we will refer to S as an L-structure. A collection of L-structures is called an L-class. Classes over singulary vocabularies are called *singulary classes*. First-order formulas defined on singulary classes are defined inductively in the usual way.

Definition: Given a singulary vocabulary L, define the syntax for L-formulas inductively as:

atomic terms	$x, y, z, \ldots$	where $x, y, z, \ldots$ are individual variables
compound terms	$f(t)$	where t is a term and f is a function symbol
atomic formulas	$P(t)$	where t is a term and P is a predicate symbol
equality	$s = t$	where s and t are terms
negation	$\neg\varphi$	where φ is a formula
implication	$\varphi \to \psi$	where φ and ψ are formulas
quantification	$\exists x\, \varphi$	where x is a variable and φ is a formula

In addition, we will use the common abbreviations for conjunction ($\wedge$), disjunction ($\vee$), and universal quantification ($\forall$). As is customary, a first-order sentence over the vocabulary L will be referred to as an L-sentence. L-sentences are Boolean queries which define properties of L-classes. The semantics of L-formulas are no different than those of formulas in ordinary first-order logic. However, it is entirely possible for a formula over a singulary vocabulary to define a binary relation, such as $\varphi(x, y) \equiv x \neq y$. In this case, the formula φ is not itself singulary. The term singulary formula will be reserved for formulas with precisely one free variable.

It is important to realize that even though equality is technically a binary relation, it is not necessary to use pairs of elements for its interpretation. This is unlike the more general case of a graph where knowledge of its edges is required, and cannot be determined by an examination of

its vertices alone. In other words, equality is not a real relation in the sense that it does not engage distinct elements of a structure. Yet its presence is required as we shall see in the next section.

§2 Singulary interpretations

In this section we show that for any vocabulary L whatsoever, there is a singulary vocabulary L^* which interprets it in a first-order way. That is to say, every L-structure S can be uniformly transformed into a singulary L^*-structure S^* so that the resulting L^*-class has essentially the same first-order properties as the original L-class. Moreover, this transformation is an elementary interpretation so that first-order L-formulas are convertible to equivalent L^*-formulas. In a certain sense, the elementarily definable properties of any class can be mirrored in a singulary class.

It is not necessary to consider vocabularies L with function symbols (or constants), since it is easy to adapt these into a purely relational vocabulary (by adding an extra place for the image). Furthermore, we can assume that all the relation symbols have the same arity (by adding dummy variables to pad up to the maximum arity if necessary). So, without loss of generality, we will assume that L is a relational vocabulary $R_1, \ldots, R_m$ of uniform arity $k \geq 2$. In this case we say that the L-structure $S = \langle A, R_1, \ldots, R_m \rangle$ is of arity k.[†] For each such structure, let $\mathrm{T}(S)$ be the set of k-tuples $\langle s_1, \ldots, s_k \rangle$ appearing in S, either in some relation $R_i(s_1, \ldots, s_k)$ or as *ground* k-tuples $\langle s, \ldots, s \rangle$ representing an original domain element s in S.

Let L^* be the singulary vocabulary $\{P_1, \ldots, P_m, f_1, \ldots, f_k\}$. To each L-structure S, associate the singulary L^*-structure S^* of size $O(|S|^k)$ by the following uniform construction.

Definition: Let $S = \langle |S|, R_1, \ldots, R_m \rangle$ be an L-structure of arity k. Define the L^*-structure

$$S^* = \langle \mathrm{T}(S), P_1, \ldots, P_m, f_1, \ldots, f_k \rangle$$

with domain

$$\mathrm{T}(S) = \overline{S} \cup R_1 \cup \ldots \cup R_m$$

where

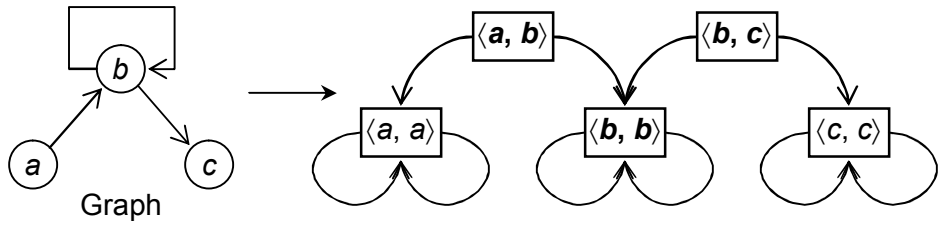
$$\overline{S} = \{\langle s, \ldots, s \rangle \in |S|^k\}.$$

Define the predicates and functions of S^* as follows:

$$P_i = \{\langle s_1, \ldots, s_k \rangle : R_i(s_1, \ldots, s_k)\} \qquad \text{for } 1 \leq i \leq m$$
$$f_j(\langle s_1, \ldots, s_k \rangle) = \langle s_j, \ldots, s_j \rangle \qquad \text{for } 1 \leq j \leq k$$

The predicates mark the nodes for each respective relation. The functions serve to project the nodes so that they can be interpreted as k-tuples – the j^{th} component of s is given by $f_j(s)$.

The simplest example is that of an ordinary directed graph, where $m = 1$ and $k = 2$. In the figure below, round vertices and straight edges are used to depict the original graph. For the singulary structure, functions f_1 and f_2 are drawn as curved arrows originating on the left and right respectively, and the predicate P_1 is demarcated using boldface inside the rectangular nodes.



Transforming a graph into a singulary structure

[†] Even though a single binary relation would suffice (as stated in Exercise 3.7 of [Immerman]) it is more instructive to examine the uniform arity case. See also previous footnote.

If we refer to each instance of $R_i(s_1, \ldots, s_k)$ as a *fact*, then we can think of the mapping into a singulary structure as a conversion in which each extant fact in the original k–ary structure is converted to an actual member of the singulary domain. In this way, the singulary image structure is a more accurate representation of the amount of information contained therein since it has one node for each fact. If each fact in the original structure necessitated some matter for storage, then the size of the singulary structure (number of nodes) would be proportional to the total mass of the original structure. Note that the number of facts in a singulary structure is precisely proportional to its mathematical size since, on a per node basis, each monadic predicate symbol contributes one bit of information, and each monadic functions symbol contributes one element.

Perhaps the ideas in this result are folklore, but it is still worth stating (and proving) in its entirety.

Theorem: Let $L = \{R_1, \ldots, R_m\}$ be a relational vocabulary of arity $k \geq 2$. Then for each L-sentence θ there is an equivalent L^*-sentence θ^* such that for all L-structures S,

$$S \models \theta \quad \Leftrightarrow \quad S^* \models \theta^*.$$

Moreover, the L^*-class $\{S^* : S \text{ is an } L\text{-structure}\}$ is an elementary class.

Proof: To see that we can translate any first-order sentence θ, we prove a stronger statement for formulas. Given $\theta(x_1, \ldots, x_l)$ for $l \geq 0$, construct $\theta^*(x_1, \ldots, x_l)$ according to the following instructions. For each variable w appearing free or bound in θ, we need to say it is a ground element. Do this by adding the following qualifier, which expresses that w is a fixed-point of every function:

$$\wedge\{f_j(w) = w : 1 \leq j \leq k\} \qquad \text{(these are the loops in the diagram).}$$

Each occurrence of an atomic formula $R_i(z_1, \ldots, z_k)$ must be replaced by a formula asserting a new variable z which represents the tuple $\langle z_1, \ldots, z_k \rangle$ satisfying the corresponding monadic predicate P_i. This can be achieved by either of the following formulas:

$$\exists z \wedge\{f_j(z) = z_j : 1 \leq j \leq k\} \wedge P_i(z)$$

$$\forall z \wedge\{f_j(z) = z_j : 1 \leq j \leq k\} \rightarrow P_i(z)$$

(it does not matter which of these is used)

Notice that the only changes are to variables and atomic formulas. Equality between variables stays the same, as do all the Boolean operations and quantification. Note that although bound variables are added, the free variables remain the same. From this conversion, it should be obvious that $\theta(w_1, \ldots, w_l)$ will satisfy the tuple $\langle s_1, \ldots, s_l \rangle$ in the original structure S just in case $\theta^*(w_1, \ldots, w_l)$ satisfies the tuple of elements $\langle \overline{s}_1, \ldots, \overline{s}_l \rangle$ in the singulary structure S^*, where each $\overline{s}_i = \langle s_i, \ldots, s_i \rangle$ is a k-tuple in $|S|^*$.

To finish, we need to design an L^*-sentence which recognizes whether a purported singulary structure is the result of the construction defined above. The ground elements must always be fixed-points of every function. Let

$$\varepsilon(x) \equiv \wedge\{f_j(x) = x : 1 \leq j \leq k\}$$

Every component of every node must be a ground element:

$$\forall x \wedge\{\varepsilon(f_j(x)) : 1 \leq j \leq k\}$$

The model must be extensional – nodes with identical images under the application of every function must be the same (i.e., two tuples with identical components must be the same tuple):

$$\forall y\ \forall z\ \wedge\{f_j(y) = f_j(z) : 1 \leq j \leq k\} \rightarrow y = z$$

Finally, every non-ground element must represent a tuple that appears in some relation.

$$\forall x\ \neg\varepsilon(x) \rightarrow \vee\{P_i(x) : 1 \leq i \leq m\}$$

Together, these sentences axiomatize $\{S^* : S \text{ is an } L\text{-structure}\}$, and the theorem is proved.

⊣

So it is seen that, with respect to definability, singulary structures are no less general than arbitrary structures. But in general, there will be an unbounded number of elements in the singulary model that reference the ground elements, leading to structures of unbounded-degree. As stated in the introduction, our real interest is in examining structures of bounded-degree, and it is to this topic that we turn to next.

§3 Bounded-degree classes

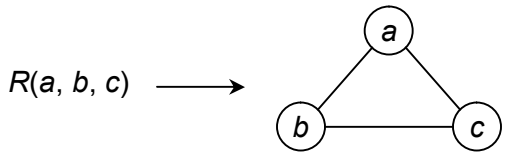
This section illustrates that passage to the singulary model does not change the notion of a class of relational structures having bounded-degree. Our first aim is to carefully define the notion of degree in either case.

The degree of a relational structure can be determined by examining its Gaifman graph, which puts an edge between a pair of nodes iff they occur jointly in a tuple of some relation.

Definition: The *Gaifman graph* of a relational L-structure S is the simple graph determined by:

$$G(S) = \langle |S|,\ \{\langle a, b\rangle : a \neq b\ \&\ S \models R(\ldots a, \ldots b, \ldots) \vee R(\ldots b, \ldots a, \ldots) \text{ for } R \text{ in } L\}\rangle$$

E.g. a single triple (of distinct elements) introduces a triangle of edges:



Edges contributed to the Gaifman graph by a 3-tuple

The degree of an element is then defined just to be its degree in the Gaifman graph.

Definition: In a relational structure S, the *degree* of a vertex s is just its degree in $G(S)$,

$$\deg(s) = |\{s' : s'\text{—}s\}| \qquad \text{where } s'\text{—}s \text{ is shorthand for an edge between } s' \text{ and } s.$$

The degree of S is then defined to be just the degree of its Gaifman graph – the supremum of degrees realized in $G(S)$.

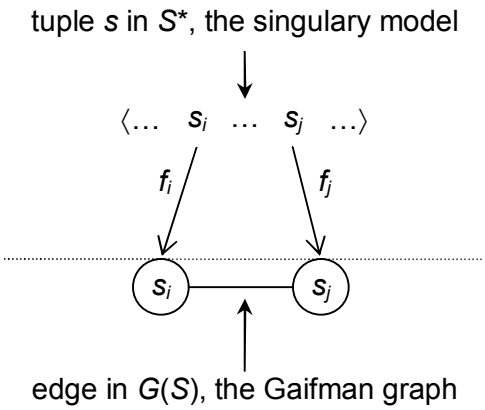
A node in a singulary structure includes both outgoing and incoming arrows. The out-degree of each node is fixed; one arrow for each of the k functions. So it makes sense to count only the in-degree of each node.

Definition: The degree of a node in a singular structure is the sum total of all arrows entering it:

$$\deg(n) = \sum|\{n' : f_j(n') = n\}|\ \ 1 \leq j \leq k$$

Note that 'parallel' arrows (corresponding to distinct functions) entering n are counted separately, and that self-loops also count. The degree of a singulary structure is also the supremum of degrees realized in it.

Below is a picture showing the relationship between S^*, the singulary structure for S, and edges in $G(S)$, the Gaifman graph for S. Note though that this figure is a *superposition*: the image of a singulary function is actually a tuple of S (consisting of the same repeated element); whereas the nodes in the Gaifman graph are actual elements (not tuples) of S.



Superposition illustrating the situation $f_i(s) \neq f_j(s)$ for $i \neq j$.

The importance of this, as we shall see, is that there is a uniform relationship between the degree of S^* and the degree of $G(S)$. So degree bounds on singulary models translate into degree bounds for Gaifman graphs, and vice versa. Indeed, the degree of an element s in the Gaifman graph $G(S)$ and the degree of its corresponding tuple $\bar{s} = \langle s, \ldots, s \rangle$ in the singulary structure S^* are correlated in a fashion that is dependent only on k, independent of S. In the proof, the key idea will be to bound the number of tuples of S that s participates in. Recall that by definition $T(S)$ contains only k-tuples that actually appear in the structure S.

Definition: Let $T(s) = \{\langle s_1, \ldots s_i, \ldots s_k \rangle \in T(S) : \text{some } s_i = s\}$ be the tuples of S that s appears in.

A class of structures is said to be of *bounded-degree* if the degree of each structure in the class is less than some fixed number. Now we are ready to state the result of this section.

Claim: Let C be a class of relational structures of arity k. Then C is of bounded-degree iff the class of singulary structures $C^* = \{S^* : S \text{ in } C\}$ is also of bounded-degree.

Proof: Consider a relational structure S of arity k. To prove the result, our aim is to show that $\deg(s)$ is bounded in $G(S)$ iff the size of $T(s)$ is bounded iff $\deg(\bar{s})$ is bounded in S^*. It will not be necessary to consider the other tuples in the singulary structure S^* because they are of degree 0.

Suppose s has degree d in the Gaifman graph $G(S)$. This means that whenever s appears in a k-tuple of S, only d other elements can appear together with s. There are at most $k(d+1)^{k-1}$ possible tuples that can be so constructed (s takes up one of the k slots, and the remaining $k-1$ other slots are filled by the d neighbors of s, including possibly s itself). So $|T(s)|$ has been bounded.

In the other direction, suppose there are l tuples in $T(s)$. By definition, in $G(S)$ every neighbor s' of s must appear in one of these k-tuples, and there are no more than $l(k-1)$ places where they can occur (the $k-1$ slots remaining in each of the l tuples). So $\deg(s)$ has been bounded in the Gaifman graph.

Now suppose that $\bar{s}$ has degree d in the singulary model S^*. This implies there are at most d tuples of S in which s appears, because every occurrence of s in a tuple (and there may be

multiple occurrences) generates its own distinct edge into $\bar{s}$. So including $\bar{s}$ itself, we have the bound $|T(s)| \leq d+1$.

Conversely, suppose there are l tuples in $T(s)$. Every arrow leading into $\bar{s}$ must come from one of those l tuples, and each one can produce at most k arrows leading into $\bar{s}$ (one for each function). This places a bound $\deg(\bar{s}) \leq l \cdot k$, which completes the proof.

⊣

So we see that the notion of a bounded-degree class of structures is invariant under interpretation as either a relational database or a singular model. Even if we were to interpret a singular model as a relational structure, this notion would stay invariant.

Consider an arbitrary singulary model $M = \langle |M|, P_1, \ldots, P_m, f_1, \ldots, f_k \rangle$, not necessarily obtained as a result of the conversion process detailed above. It is also possible to convert M into a relational structure $M' = \langle |M|, P_1, \ldots, P_m, R_1, \ldots, R_k \rangle$, replacing each function f_i by a binary relation $R_i = \{\langle a, b \rangle : f_i(a) = b\}$ representing its graph. Furthermore, the Gaifman graph of M' will just be the symmetric irreflexive closure of the union of all the R_i. It should be obvious that its degree is essentially the same as the degree of M.

At this point, it may not be clear why feasible models of information need to have bounded-degree. Any physically instantiated relational database must be stored as data in memory. Regardless of the sophistication or technology used, individual datum must occupy space and be somehow connected to one another. When put in these terms, the appropriate model for physically stored information becomes clearer.

§4 Singulary models for Data structures

Data structures are standard models of how information is stored in a computer – finite arrangements of nodes containing data, each linked to other nodes by pointers. It provides a mechanism of dynamically increasing information storage capacity by simply increasing its size (the number of nodes), keeping the amount of information contained in each node fixed, thereby providing a uniform method of storing and accessing the data. Although the structure (pattern of links) can be arbitrary, the amount of Boolean data stored at each node is fixed, as well as the number of outgoing pointers. Common examples include lists (linear sequential arrangement of nodes), trees, and more generally graphs of fixed out-degree. Each node can be visualized as a fixed-width container holding bits and addresses:

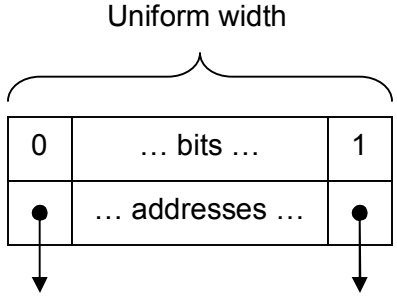


Figure illustrating one node in a data structure

When a particular pointer is not being used, it is given the value *nil*, which stands for 'nothing'. In figures this is indicated by a dot, without the customary arrow extending from it.

It is easy to see that singulary structures provide a perfect model for data structures. Each bit corresponds to the Boolean value of a predicate, and each address to the value of a function (nil values point to a common 'bottom' $\perp$).

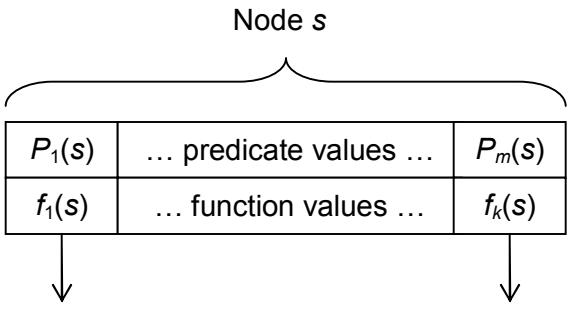


Figure illustrating an element in a singulary model

Doubly-linked Data Structures

Extracting information from a data structure via an algorithm requires navigating the links. The easiest way to do this is when the structure is 'doubly-linked' – each link is provided with a return link, providing a convenient way to reverse direction and back out of a node. This also forces the data structure to be of bounded-degree, since the number of incoming links must not exceed the number of outgoing links, which are fixed by design.

Perhaps there is a more significant lesson to be learned from considering the physical necessity of storing the data structure in a real memory. It is inconceivable that an unbounded number of memory cells could all reference the same location in constant time. For this would mean they would all have to be within a bounded distance of each other – an untenable situation in finite-dimensional space. So although it might be theoretically possible for an unbounded number of locations to point at a given location, there can be no practical method of executing those memory references directly without impossible overcrowding. Introducing "way stations" would be a method of indirection that attempts to solve the problem by navigating through a bounded-degree substructure. A much more substantial explanation of these concepts and their significance to logic can be found in [Lindell].

If only a bounded number of links can enter a node, then they might as well be reversible (remember that directions in space are inherently reversible), in which case the corresponding functions in the singulary model will be invertible. This construction is laid out in the appendix. So clearly, the natural model for doubly-linked data structures is to use a singulary vocabulary which has an inverse for each function.

Definition: An *invertible* singulary vocabulary L contains an inverse for each function symbol.

$$L = \{P_1, \ldots, P_m, f_1, \ldots, f_k, f_1^\circ, \ldots, f_k^\circ\}$$

We are using f° to denote the syntactic inverse of f, to distinguish it from the semantic inverse, and to indicate that it is a separately interpreted symbol (which happens to always be the inverse). When viewed as a graph, each bijective pair (f, f°) determines edges which partition the finite domain into a collection of disjoint cycles.

A problem appears to arise in the case of nil values, because it is not possible for $\perp$ to accommodate an arbitrarily large number of incoming arrows. But these are easily dealt with as illustrated in the following figure, entirely obviating the need for $\perp$.

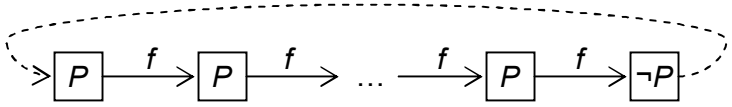


Figure illustrating wraparound

The dotted line represents a value introduced to replace nil, ensuring totality for f. It is determined uniquely by tracing backwards along the value chain for f (always possible since f is injective). This procedure terminates because every data structure has a finite domain, and the final value realized is the one desired. The only thing left is to introduce a new predicate symbol P whose sole purpose is to indicate the endpoints where dotted lines were employed. This enables recovery of the original partial function via $f(x) = \perp$ iff $\neg P(x)$.

The real value of viewing doubly-linked data structures as singular models will become apparent in the next section, where the mathematical elegance of these invertible vocabularies bears fruit by allowing first-order formulas to collapse into a much simpler form.

§5 Normal forms

In earlier sections we saw that elementary definability over arbitrary vocabularies is essentially the same over singulary vocabularies. However, the singulary structures involved will in general have unbounded degree. We also saw that the notion of a bounded-degree class of structures over arbitrary vocabularies is essentially the same over singulary vocabularies. A class of singulary structures is guaranteed to have bounded-degree if every function symbol has a corresponding inverse (indeed, this guarantees regularity – every node will have the same degree). Moreover, data structures that result from physically embedding data into storage media are naturally doubly-linked, and easily modeled within the realm of invertible singulary vocabularies. Under these special circumstances, elementary definability takes on a particular elegance. First-order formulas can always be written in a special normal form where quantifiers are never nested. To demonstrate this we introduce a special type of threshold quantifier which generalizes in a natural way ordinary existential quantification.

Definition: For each natural number n, the *numerical quantifier* $\exists^n x...$ stands for "there are (at least) n distinct x's satisfying …". In case $n = 1$, this is the ordinary existential quantifier.

Obviously, numerical quantifiers do not extend the realm of first-order definability because for any fixed n, $\exists^n$ can be rewritten as a combination of n ordinary existential quantifiers.

$$\exists^n x\ \omega(x) \quad \equiv \quad \exists x_1 \ldots x_n\ \{x_i \neq x_j \colon 1 \leq i < j \leq n\} \wedge \{\omega(x_l) : 1 \leq l \leq n\}$$

But these quantifiers do serve to simplify the statement and proof of our normal form theorem, which says that over any class of (finite or infinite) structures in an invertible singulary vocabulary, every first-order formula can be rewritten as a Boolean combination of quantifier-free formulas together with sentences composed of numerical quantifiers applied to quantifier-free formulas. In particular, the quantifiers are never nested.

Theorem: Let $\zeta(\bar{x})$ be a first-order formula in an invertible singulary vocabulary. Then $\zeta(\bar{x})$ is equivalent to a Boolean combination of the following:

Atomic formulas:	$\alpha(\bar{x})$	&
Numerical quantifier sentences:	$\exists^n x\ \beta(x)$	where β is quantifier-free.

Proof: By induction on the ordinary first-order syntax – all cases except quantification are trivial. Consider $\exists x\ \varphi(x, \bar{y})$ where φ is in the normal form as specified in the statement of the theorem. By putting φ in disjunctive normal form, distributing the existential quantifier over the disjunctions, and removing any sentences from underneath the scope of quantification, it suffices without loss of generality to consider φ as a conjunction of atomic clauses (or their negations). In a singulary environment, clauses involving a predicate symbol contain only one variable. So the only way that variables can be related to each other is through the equality symbol. Each

equation relates just two variables (not necessarily distinct), one occurrence on each side of the equals sign.

If a variable has no connection to x via equations in φ, it should be removed. Say that two (distinct) variables are *related* if they occur jointly in an equation together, and *connected* if they are in the (reflexive) transitive closure of this symmetric relation (i.e. directly or indirectly related). All variables not connected to x can be factored out from under the scope of the quantifier $\exists x$. To see how to do this, take any quantifier-free conjunctive formula $\theta(x, \bar{y}, \bar{z})$ in which x is connected to every y in $\bar{y}$ and no z in $\bar{z}$. Since it is impossible for any z to occur in the same clause with x or any y, we can simply separate out all clauses involving variables from among the $\bar{z}$ as follows:

$$\exists x\, \theta(x, \bar{y}, \bar{z}) \qquad \text{becomes} \qquad \phi(\bar{z}) \wedge \exists x\, \varphi(x, \bar{y})$$

So without loss of generality, we can assume that each y in $\varphi(x, \bar{y})$ is connected to x.

If there are no $\bar{y}$, we are done (for the sentence $\exists x\, \varphi(x)$ is already in the correct form). Otherwise x must participate in at least one nontrivial equation (i.e. an equation involving another variable y in $\bar{y}$), which must be of the form $G(x) = H(y)$, where G and H are compositions of the function symbols. These can always be solved for x by utilizing the critical assumption of bijectivity to rearrange the terms:

$$G(x) = H(y) \qquad \text{becomes} \qquad x = G^{\circ}H(y)$$

where G° stands for the inverse sequence of functions appearing in G, a crucial step that is only possible in an invertible singulary vocabulary. So assume without loss of generality that whenever x appears in a nontrivial equation from φ, it is already in the form $x = F(y)$ or $x \neq F(y)$, where F stands for any composition of functions. A positive equation $x = F(y)$ is called an *identity*, whereas a negative equation $x \neq F(y)$ will be called a *diversity*.

After solving for x, there are two cases depending on the polarity of its equations.

Case 1: If an identity $x = F(y)$ appears in φ for some y in $\bar{y}$, just substitute the term $F(y)$ for x everywhere and eliminate the quantifier:

$$\exists x\, \varphi(x, \bar{y}) \qquad \text{becomes} \qquad \varphi(F(y), \bar{y})$$

where $\varphi(F(y), \bar{y})$ is the formula resulting from substituting $F(y)$ for x everywhere in $\varphi(x, \bar{y})$. Observe that the resulting formula is of the correct form (i.e. it is quantifier-free). Clearly, if φ asserts that $x = F(y)$, then $\exists x\, \varphi(x, \bar{y})$ can only be true provided $\varphi(F(y), \bar{y})$. On the other hand, if $\varphi(F(y), \bar{y})$ is satisfied, then $\exists x\, \varphi(x, \bar{y})$ is satisfied by $x = F(y)$.

Case 2: Otherwise, every nontrivial equation in φ involving x must be a diversity $x \neq F(y)$. The remaining occurrences of x in φ must be single variable occurrences. These properties of x fall into two types of atomic forms: positive or negative predicates $P(F(x))$ or $\neg P(F(x))$; and trivial equations $x = F(x)$ or $x \neq F(x)$ representing closed or open paths from x respectively. No matter what form they take, collect all these properties together into a formula γ:

$$\gamma(x) \equiv \bigwedge \{\alpha(x) : \alpha(x) \text{ is a clause in } \varphi(x, \bar{y})\}$$

Now we can rewrite φ:

$$\varphi(x, \bar{y}) \qquad \text{becomes} \qquad \gamma(x) \wedge \psi(x, \bar{y})$$

where the only clauses in ψ involving x are diversities $x \neq F(y)$. As observed previously, x must relate to at least one y in $\bar{y}$. In order for x to exist, it must both satisfy γ and be distinct from any possible "competitors" $F(y)$ which may happen to also satisfy γ. E.g., if none of the $F(y)$ satisfy γ, then we are done, as $\exists x\, [\gamma(x) \wedge \psi(x, \bar{y})]$ is equivalent to $[\exists x\, \gamma(x)] \wedge \psi'(\bar{y})$, where ψ' is $\psi(x, \bar{y})$ in

which all appearances involving x (i.e. diversities) have been deleted. More generally, to take into account all possibilities, consider any subset T of terms $F(y)$ diverse from x:

$$(*) \qquad T \subseteq \{F(y) : x \neq F(y) \text{ is in } \psi\}$$

This circumstance, of $x \neq T$, can be pictured as follows (in the case that T is not empty):

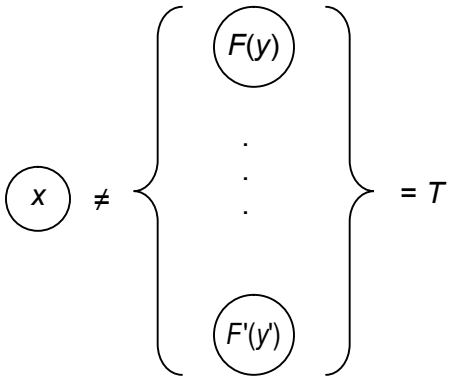


Figure illustrating situation where an element x is diverse from the terms in T.

No term in T can be in competition with x unless it satisfies γ. So let the formula $\gamma[T]$ say that each of these terms satisfy γ:

$$\gamma[T] \equiv \wedge\{\gamma(F(y)) : F(y) \in T\}.$$

Note $\gamma[\varnothing]$ is vacuously true.

The size of T is a reliable indicator of the number of possible competitors only if the terms in T are distinct from one another. So let $\delta[T]$ be the formula that says these terms are distinct:

$$\delta[T] \equiv \wedge\{F(y) \neq F'(y') : F(y) \,\&\, F'(y') \text{ are distinct terms in } T\}.$$

Again, $\delta[\varnothing]$ is vacuously true.

Claim that the following conjunction of formulas $\zeta(\bar{y})$ is equivalent to the original $\exists x\, \varphi(x, \bar{y})$.

$$\zeta(\bar{y}) \equiv \wedge\{\delta[T] \wedge \gamma[T] \rightarrow (\exists^{n+1} z)\, \gamma(z) : \text{where } n = |T|, \text{ and } T \text{ ranges over } (*)\}.$$

This says that there are more elements satisfying γ than the number of distinct terms in T that also satisfy γ. Clearly, ζ is of the correct form (a combination of quantifier-free formulas or numerical quantifier sentences). It remains to be seen that $\zeta(\bar{y})$ is semantically equivalent to $\exists x\, \varphi(x, \bar{y})$. But this is pretty obvious. If $\gamma(x) \wedge \psi(x, \bar{y})$ is satisfied by some element x in a situation where the n distinct elements $F(y) \neq x$ of T also satisfy γ, then there are indeed more than n elements satisfying γ. On the other hand, ζ says that every possible collection of n distinct elements $F(y) \neq x$ satisfying γ always leaves room for another element (namely x) which satisfies γ.

⊣

Note: In the statement of the theorem, the quantifier-free component α can contain any atomic formula of singulary logic, but the sentential component $\beta(x)$ contains but a single free variable and is therefore limited to being composed of properties of paths from x: $P(F(x))$ or $x = F(x)$. This fact will be used in the following section on efficient computations.

§6 Linear-time computability

An important consequence of the normal form theorem is a linear-time algorithm for evaluating any first-order formula in an invertible singulary vocabulary. Although essentially the same as the pioneering result of [Seese] for first-order sentences over bounded-degree graphs, we are able to improve upon it by extending its reach to formulas in a unique way, by attempting to give a convincing analysis that not only sentences, but formulas with any number of free variables can be realistically evaluated in linear-time, despite the apparent contradictions in output size.

Ordinarily, evaluating a first-order formula $\sigma(\bar{x})$ over a finite model M involves explicitly computing the relation determined by σ as a query:

$$\sigma^M = \{\bar{m} : M \models \sigma(\bar{m})\}$$

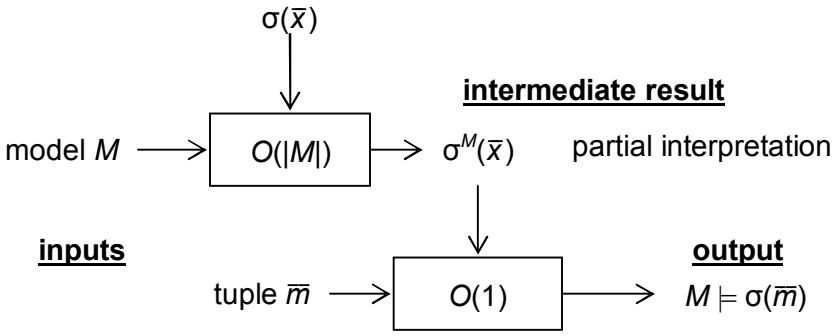
When σ is a sentence, the outcome is either *true* or *false*, and the statement that σ is computable in linear-time over invertible singulary vocabularies makes perfect sense. But even simple binary queries like $\sigma(x, y) \equiv x \neq y$ which are "easy" to compute produce super-linear size answers. Obviously, when $\sigma(\bar{x})$ has multiple free variables we cannot hope to compute the entire relation determined by σ^M in linear-time because its size (the number of tuples) is potentially polynomial in the size of M, with degree equal to the arity of $\bar{x}$.

A solution around this problem is to compute a *partial interpretation* $\sigma^M(\bar{x})$ which, when given values $\bar{m}$ for the free variables $\bar{x}$, can answer in *constant-time* whether or not $M \models \sigma(\bar{m})$. We capture this formally as a notion of implicit linear-time computability.

Definition: An L-query $q(\bar{x})$ is said to be *implicitly computable* in linear-time if for every finite L-structure S, there is a partial interpretation $q^S(\bar{x})$ such that:

1.	Given $\bar{x}$,	$q^S(\bar{x}) \Leftrightarrow S \models q(\bar{x})$	can be computed in $O(1)$ time
2.	Given S,	$q^S(\bar{x})$	can be computed in $O(\|S\|)$ time.

For a formula $\sigma(\bar{x})$ this will mean that for each M, we will compute in time linear in M another formula $\sigma^M(\bar{x})$ which demonstrates that we *implicitly* know which tuples satisfy σ over M even though we cannot list them *explicitly*. (This is no different than the trick used to compose space-bounded algorithms in which the intermediate result exceeds the allotted bound.) The concept behind this is illustrated in the figure below.



Method of partial interpretations used to implicitly calculate a query

Using this concept of implicit evaluation, we can show that the arity of a formula in singular logic does not fundamentally affect the efficiency of its evaluation.

Corollary: Let $\phi(\bar{x})$ be a first-order formula in an invertible singulary vocabulary L. Then $\phi(\bar{x})$ is implicitly computable in linear-time.

Proof: By the main theorem, $\phi(\bar{x})$ can be written in normal form as a Boolean combination of sentences and quantifier-free formulas. Given an L-structure S, each of the constituent sentences $\exists^k x\ \beta(x)$ can be explicitly evaluated in linear-time by simply scanning each node s of S to see if it satisfies $\beta(s)$. Checking $\beta(s)$ involves following bounded-length paths from s of the form $F(s)$ to see if they are closed or open (equal or not equal to s), and possibly checking a predicate P there. This requires $O(1)$ time for each s, together with counting the results. By substituting those Boolean valuations into the original formula $\phi(\bar{x})$, the resultant quantifier-free formula $\phi^S(\bar{x})$ is the desired partial interpretation. Given values $\bar{s}$ for its free variables, ϕ^S requires only $O(1)$ time to answer whether or not $\phi^S(\bar{s})$, which is obviously equivalent to $S \models \phi(\bar{s})$.

⊣

To apply this result to bounded-degree graphs, note that conversion to an invertible singulary structure can be accomplished in linear-time, as explained in the appendix.

Conclusion

When using logic to describe the properties of mathematical structures, it is taken for granted that variables can be combined into tuples. Indeed, the mathematical structures are themselves composed of relations involving tuples of elements. We have described singulary structures, a situation in which no tuples are used. These structures correspond directly to the data structures of Computer Science, where each node is of uniform capacity to store information. Although every mathematical structure, without exception, can be transformed into a singulary structure, it is necessary to admit elements of arbitrarily high degree.

If these structures were stored in memory, occupying material locations in space, there would be a natural physical bound on the amount of matter that occupied any given place. Hence, physically scalable classes of structures should have bounded-degree. For a data structure, this corresponds to being doubly-linked, and we saw that invertible singulary structures are almost ideal models of this situation. Our normal form theorem applies to these symmetrically linked models, showing that first-order definability is much simpler than it appears. In particular, quantifiers need not be applied to formulas with more than one variable, permitting a very straightforward linear-time evaluation algorithm. For future research, perhaps similarly strong normal forms can be obtained for monadic fixed-point logic over invertible singulary vocabularies.

It may appear that our results apply to only a very specialized situation, doubly-linked data structures. But we are trying to argue that these particular models are *sine qua non* to the whole notion of an asymptotically realizable class of finite structures. For how can one speak of an actual class of structures being feasible if there is no physically resourceful method that realizes, at least in principle, each structure in the class? With a natural notion of bounded-degree built in, invertible singulary vocabularies provide a compelling way to model feasible classes of structures. These "models of information" have the desirable (indeed requisite) property that each datum contains a fixed amount of information, and can access (and is accessible from) only a fixed number of locations.

Although we have emphasized the importance of physical feasibility, our analysis here is far from the last word on this subject. A more complete investigation can be found in [Lindell], where the finite models in a physically feasible class are embedded into a single infinite background structure (the universe), and functions are assigned to directions in space (instead of being arbitrary pointers) so that they are torsion-free (assuming that space extends indefinitely in every direction). However, since the results in this paper apply to the more general situation of arbitrary doubly-linked data structures, and avoid the extra complication of embedding, we have chosen to present them this way.

Acknowledgements

Help in preparing this manuscript for publication came from my wife, Suzanne Lindell, as funded by Haverford College. Her assistance, with support from the College, is greatly appreciated. I also wish to thank the Newton Institute for Mathematical Sciences for their support in visiting Cambridge, England to give a lecture based on this paper.

References

[Church, A.] *Introduction to Mathematical Logic*, Princeton University Press, 1956.

[Durand, A. & Grandjean, E.] 'First-order queries on structures of bounded degree are computable with constant delay' ACM Transactions on Computational Logic, to appear.

[Gaifman, H.] 'On local and non-local properties' in *Proceedings of the Herbrand Symposium: Logic Colloquium '81*, (J. Stern ed.) pp. 105 – 135 (North-Holland, Amsterdam 1982).

[Grandjean, E.] 'Invariance properties of RAMs and linear time' Computational Complexity, 4:62-106, 1994.

'Linear time algorithms and NP-complete problems' SIAM Journal on Computing, 23(3):573-597, 1994.

[Grandjean, E. & Schwentick, T.] 'Machine-independent characterizations and complete problems for deterministic linear time' SIAM Journal on Computing, 32(1):196-230, 2002.

[Hanf, W.] 'Model-theoretic methods in the study of elementary logic' in *The Theory of Models* (J. Addison, L. Henkin and A. Tarski, eds.) pp. 132 – 145 (North Holland, Amsterdam 1965).

[Immerman, N.] *Descriptive Complexity*, Springer, New York 1999.

[Libkin, L.] *Elements of Finite Model Theory*, Springer, 2005.

[Lindell, S.] 'An elementary term logic for physically realizable models of information', in *The Old New Logic: Essays on the Philosophy of Fred Sommers*, David Oderberg editor, MIT Press, 2005.

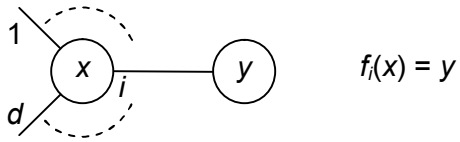
[Quine, W.V.] *Mathematical Logic*, W.W. Norton & company, 1940.

[Seese, D.] "Linear time computable problems and first-order descriptions" in Mathematical Structures in Computer Science, 6(6):505-526, December 1996.

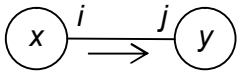
Appendix

Summary
Here we illustrate the precise connection between bounded-degree (simple) graphs, doubly-linked data structures, and invertible singulary structures. The goal is to get from a simple graph of degree d to an invertible singulary structure in time $O(n)$, where n is the number of nodes. The key technique is to use a local ordering of the outgoing and incoming edges at each node.

Preliminaries
In a simple graph of degree d, arbitrarily number the edges at each vertex 1, …, d to make them into (outgoing) arrows $f_1, \ldots, f_d$ for a data structure. I.e. $f_i(x) = y$ if from x there is an edge labeled i which leads to y. Set $f_i(x) = x$ when there is no i^{th} edge (i.e. a *nil*).

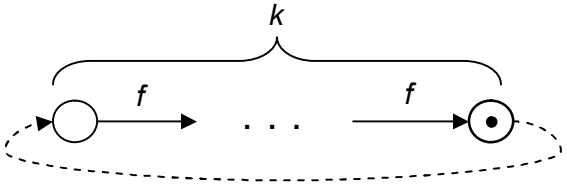


$E(x, y) \equiv \vee \{f_i(x) = y : x \neq y,\ 1 \leq i \leq d\}$ will define the original edge relation of the graph. The result is a doubly-linked data structure, in the sense that for each link $x \rightarrow f_i(x) = y$, there is a reverse link $f_j(y) = x \leftarrow y$ (for nils, $i = j$). But the functions themselves might not be injective. To make (partial) injective functions, let $f_{i,j}(x) = y$ if an edge leaving x at position i enters y at position j.



Clearly, each $f_{i,j}$ is one-one, and $f_{i,j}^{-1} = f_{j,i}$ except where it is undefined. This creates a doubly-linked data structure in which there is a uniform correspondence between each link and its reverse, even though there are a lot of nil values (places where $f_{i,j} = \perp$). The original data structure, modeled by functions f_i where i refers to the exit position only, can be recovered by letting $f_i(x) \equiv f_{i,j}(x)$ for the unique j for which $f_{i,j}(x)$ is defined.

Final step
(This is also discussed in the main body of the paper.) Each partial injective function can be made total (and hence surjective) by *wraparound*: if $f(x) = \perp$, then define $f(x) = f^{-k}(x)$ for the unique integer $k \geq 0$ such that $f^{-k-1}(x) = \perp$. This takes linear time to process, amortized over all nodes. It is necessary to add a predicate at the front to record the original nil value, (which could also be used if desired to allow for self loops in the original graph).



Each constructed bijective function is necessarily a permutation – a collection of disjoint cycles in a finite structure – which leads directly into the definition of invertible singulary structures.

Note
This will also work for directed graphs (data structures of bounded indegree and outdegree) by using additional predicates to mark the reverse edge $f_{j,i}$ as a "fake".