The Expressive Power of Unary Counters

Michael Benedikt and H.J. Keisler

Expressive Power of Unary Counters

Michael Benedikt[1] **H. Jerome Keisler**[2]

[1] Bell Laboratories, 1000 East Warrenville Rd., Naperville, IL 60566, USA, Email: benedikt@bell-labs.com
[2] University of Wisconsin, Madison Wisconsin 53706, Email: keisler@math.wisc.edu

Abstract. We compare the expressive power on finite models of two extensions of first order logic L with equality. $L(Ct)$ is formed by adding an operator $count\{x : \varphi\}$, which builds a term of sort $\mathbf{N}$ that counts the number of elements of the finite model satisfying a formula φ. Our main result shows that the stronger operator $count\{t(x) : \varphi\}$, where $t(x)$ is a term of sort $\mathbf{N}$, cannot be expressed in $L(Ct)$. That is, being able to count elements does not allow one to count terms.
This paper also continues our interest in new proof techniques in database theory. The proof of the unary counter combines a number of model-theoretic techniques that give powerful tools for expressivity bounds: in particular, we discuss here the use of indiscernibles, the Paris-Harrington form of Ramsey's theorem, and nonstandard models of arithmetic.

1 Introduction

Most database query languages are based on some version of first-order logic. However, practical query languages such as SQL generally supplement their pure first-order component with certain primitives, among them the ability to count over the database. An active line of research in database theory has been to model the impact of counting on a database language by studying extensions of first-order logic by *counting quantifiers* [16] [10]. The goal is to characterize the expressive power of various counting languages, and to identify those with and without good analytical and computational properties.

In [16] logics with n-ary counters $FO + C^n$ are introduced. A method of proving upper bounds on the complexity of these languages is introduced, relying on an Ehrenfeucht-Fraisse game construction for counting (see also [17]). This technique is used to prove a hierarchy theorem for this sequence of logics, in the case of unnested counters. The games are also exploited in [9] and [10] to get complexity bounds for languages allowing counting quantifiers of the form $\exists i$.

As opposed to the works cited above, we will consider languages with counting constructs that can interact with arbitrary expressions over the integers. We shall consider three extensions of first order logic formed by adding term-building operators which count the number of elements satisfying a formula. The smallest of these is a first-order logic with unary counters, like the language $FO + C^1$ considered in [16], but with arbitrary integer predicates applicable to the counters. This language is more similar in spirit to the rich two-sorted languages considered in [13] than to the more restricted ones of [10] and [9]. Although this language is extremely large, we will show some interesting limits on its expressive power by displaying two counting languages with more expressive power. In particular, we will show that it is impossible to count the number of equivalence classes of a binary relation in the language of unary counters, and it is impossible to count the number of connected components of a graph. Our results also serve to show that the ability to count the number of elements satisfying a property does not suffice to count the number of terms.

In the process, we will introduce modifications to the Ehrenfeucht-Fraisse argument that we think are interesting in its own right. We will make use of two model-theoretic techniques: nonstandard universes and indiscernibles. Indiscernibles and nonstandard models have shown up either implicitly or explicitly in several recent works in database theory [26] [4]. These techniques were recently used to settle a number of other problems concerning the expressive power of query languages [3], and they can be used to simplify the bookkeeping involved in many Ehrenfeucht-Fraisse arguments. We believe these techniques can be particularly helpful for proving expressivity bounds for languages involving aggregates.

As mentioned before, one of the principal reasons for studying the expressive power of query languages is to give insight into the design of languages with desirable properties. At the end of this paper we will apply our main theorem to show that a particular desirable property—the weakest precondition property—fails for a natural database language with unary counters.

Organization: Section 2 gives the definition of the languages we will deal with in this paper. Section 3 describes the nonstandard framework we use to analyze expressivity of queries, and gives some introductory examples of its usefulness. Section 4 outlines the proof of the main result. Section 5 gives a version of this result for a language similar to the tuple relational calculus with range-restriction, and gives an application of these results to the weakest precondition problem for this language.

2 Preliminaries

We give the notation for first-order logic and several counting languages.

Let L be a first-order language with equality and finitely many relation, function and constant symbols. Let $\mathbb{U}$ be an infinite set. When we talk about *finite L-structures*, we will mean structures whose domain is a finite subset of $\mathbb{U}$. Let Λ be a countable set of relations and functions on the set $\mathbf{N}$ of natural numbers which contains at least equality and a constant for each $n \in \mathbf{N}$, and remains fixed throughout our discussion.

Our first extension of L, denoted by $L(Ct)$, adds to L the term building operator

$$count\{x : \varphi(x, \ldots)\}$$

which bounds the variable x, and has a symbol for every element of Λ. The models for $L(Ct)$ are finite structures $\mathcal{A} = \langle A, \ldots \rangle$ with vocabulary L, and the count operator is interpreted in $\mathcal{A}$ as the cardinality of the set of all elements $x \in A$

which satisfy the formula φ in $\mathcal{A}$. Similarly, the term $\tau(\vec{y}) = count\{x : \varphi(x, \vec{y})\}$ defines a function that for each $\vec{y}$ returns the cardinality of elements in $\mathcal{A}$ satisfying $\varphi(x, \vec{y})$.

More precisely, the language $L(Ct)$ has terms of the two sorts U and $\mathbf{N}$. The terms of sort U are the same as for first order logic; in particular, the variables are of sort U. $L(Ct)$ has the usual first order rules for building formulas, plus

- For each formula φ and variable x, $count\{x : \varphi\}$ is a term of sort $\mathbf{N}$.
- If $t_1, \ldots, t_n$ are terms of sort $\mathbf{N}$, then $f(t_1, \ldots, t_n)$ is a term of sort $\mathbf{N}$, and $r(t_1, \ldots, t_n)$ is a formula for each n-ary function $f \in \Lambda$ (relation $r \in \Lambda$).

Our second extension of L, denoted by $L(Tm)$, adds to $L(Ct)$ the more general term-building operator

$$count\{t(x, \ldots) : \varphi(x, \ldots)\}$$

where φ is a formula and $t(x, \ldots)$ is a term of sort $\mathbf{N}$. In a model $\mathcal{A}$, it counts the number of distinct values of $t(x, \ldots)$ such that x satisfies φ in $\mathcal{A}$. Thus, we always have $count\{t(x, \ldots) : \varphi(x, \ldots)\} \leq count\{x : \varphi(x, \ldots)\}$.

Our third extension, $L(Ct, \mathbf{N})$, adds to $L(Ct)$ variables of sort $\mathbf{N}$ and quantifiers over variables of sort $\mathbf{N}$. The language $L(Ct, \mathbf{N})$ was considered in [16]. The main object of study in that paper was a language like $L(Ct, \mathbf{N})$ which had counts over n-tuples of variables of sort U, but did not allow nesting of counts.

In this paper we shall show that $L(Tm)$ and $L(Ct, \mathbf{N})$ are proper extensions of $L(Ct)$.

We remark that quantifiers of sort U can be eliminated from each of the languages $L(Ct)$, $L(Tm)$, and $L(Ct, \mathbf{N})$. An existential quantifier can be eliminated by replacing a formula $\exists x \varphi(x, \ldots)$ by $\neg count\{x : \varphi(x, \ldots)\} = 0$.

In a vocabulary L with function symbols, one might also consider the term builder $count\{s(x, \ldots) : \varphi(x, \ldots)\}$, where the term s is of sort U. This term builder is already definable in $L(Ct)$, because the equation

$$count\{s(x, \ldots) : \varphi(x, \ldots)\} = count\{z : \exists x(z = s(x, \ldots) \wedge \varphi(x, \ldots))\}$$

holds in all finite models.

3 Model-theoretic techniques and expressive bounds

We discuss here some model-theoretic techniques that are helpful for giving expressive bounds, and which will be used in the main result of Section 4.

3.1 Nonstandard Models

The work in [3] made use of nonstandard models and indiscernibles as techniques for analyzing expressivity bounds. Here we discuss them in more detail.

The naive approach to showing that a property Q is not expressible in some language $\mathcal{L}$ is to get two models that agree on all sentences of $\mathcal{L}$, but disagree on

Q. The problem immediately encountered in applying this technique in finite-model theory is the following: Any two finite models which satisfy the same sentences of a first order language L are isomorphic, and thus satisfy the same sentences of any reasonable logic, including $L(Ct)$, $L(Tm)$, and $L(Ct, \mathbf{N})$. The standard technique for circumventing this problem is via Ehrenfeucht-Fraisse games ([8], [7]). One decomposes the sentences of the logic into countably many fragments $\mathcal{L}_n$, and then constructs for each n two finite models M_n and N_n agreeing on fragment $\mathcal{L}_n$ but disagreeing on Q.

Here, we give an alternative to this construction. Inexpressibility bounds are obtained by finding two hyperfinite (meaning, informally for now, "infinitely large finite") models M and N agreeing on all queries in $\mathcal{L}$, but disagreeing on Q. The first virtue of this technique is as a way of abstracting away from the bookkeeping involved in Ehrenfeucht-Fraisse constructions. For example, if one is interested in showing the inexpressibility of connectivity within pure first-order logic, one need only look at the two hyperfinite graphs G_1 and G_2, where G_1 is a single hyperfinite cycle, while G_2 is the union of two hyperfinite cycles. A single game argument shows these two to be elementarily equivalent in first-order logic, but only one is connected, hence connectivity is not first-order definable.

The above example may appear to make the technique of nonstandard models useful more as a convenience than as an essential tool. However, the technique becomes particularly useful when dealing with expressivity results for higher-order logics. We will defer a more detailed discussion of the use of nonstandard models in higher-order languages to the full paper. The use of nonstandard universes (as opposed to elementary extensions) allows one to work with hyperfinite extensions of traditional aggregates constructs such as Counts, Sums, and set formers. For example, consider the query Q over complex objects (see [5],[25],) asking whether a given set of sets A contains two sets of differing parity. One can show this query to be inexpressible in the nested relational algebra [5], a higher-order analog of the relational algebra, by considering a natural counterexample: two hyperfinite structures S_1 and S_2, the first consisting of two hyperfinite sets of differing parity, the second consisting of two hyperfinite sets of the same parity. This argument can be formalized in a straightforward way using the definitions below, and can be easily generalized to higher-order queries. Arguments such as this are difficult to prove using other techniques (see the discussion in [20]) and can't be formalized using the 'flat' ultraproduct or elementary extension constructions. On the other hand, direct constructions using ultraproducts, when available, are often more concrete and more accessible in terms of exposition than the use of a nonstandard universe(compare, for example [3] and [24]).

In the previous paragraph, we spoke informally of hyperfinite structures. We now give some formal definitions, following the exposition in [3]:

For any set S, the **superstructure** $V(S)$ over S is defined as $V(S) = \bigcup_{n<\omega} V_n(S)$ where $V_1(S) = S$, and $V_{n+1}(S) = V_n(S) \cup \{X \mid X \subset V_n(S)\}$.

We will work with the structure $\langle V(S), \in \rangle$ considered as a structure for the first-order language for the epsilon relation. A **bounded-quantifier formula** in this language is a formula built up from atomic formulas by the logical connectives and the quantification: $\forall X \in Y$, $\exists X \in Y$, where X and Y are variables.

A **nonstandard universe** consists of a pair of superstructures $V(S)$ and $V(Y)$ and a mapping $* : V(S) \to V(Y)$ which is the identity when restricted to S (i.e. ${}^*x = x$ for each x in S) and which satisfies

1. $Y = {}^*S$.
2. *(Transfer Principle)* For any bounded quantifier formula $\phi(v_1, \ldots, v_n)$ and any list $a_1, \ldots, a_n$ of elements from $V(S)$, $\phi(a_1, \ldots, a_n)$ is true in $V(S)$ if and only if $\phi({}^*a_1, \ldots, {}^*a_n)$ is true in $V(Y)$.

An element of $V(Y)$ is **standard** if it is in the image of the $*$-map. An element of $V(Y)$ is **internal** if it is in the downward transitive closure of the standard sets under $\in$. Elements of $V(Y)$ that are not internal are called **external**. An internal map is a map whose graph is an internal set.

We will assume that our universe also satisfies the following

3. *(Countable Saturation Principle)* For every standard A, and every countable collection $\Sigma(x, \vec{v})$ of bounded-quantifier formulas, and for every vector $\vec{c}$ of internal sets, if every finite subset of $\Sigma(x, \vec{c}/\vec{v})$ is satisfied in $V(Y)$ by some element of A, then $\Sigma(x, \vec{c}/\vec{v})$ is satisfied by an element of A.

We often omit the $*$ when convenient: for example, if $<$ is an ordering on a set A, and $x_1, x_2 \in {}^*A$, then we will write $x_1 < x_2$ rather than $x_1 {}^*\!< x_2$. If Q is a query on schema SC, and M is a *-database (an element of the set ${}^*DB$, where DB is the set of all SC-databases), then we will refer to $Q(M)$ rather than ${}^*Q(M)$.

For our proofs we take the base set S of $V(S)$ to be the disjoint union of the domain $\mathbb{U}$ from which our finite structures are taken and $\mathbb{N}$.

By a ***-finite set** we mean any set in the *-image of the collection of finite subsets of some standard set. Equivalently, an internal set B is *-finite if there is an internal bijection of B onto an initial segment of ${}^*\mathbb{N}$. In particular, we can talk about *-finite structures, which will have their underlying domains being contained in the *-image of the finite powerset of $\mathbb{U}$. By the transfer principle, such sets B have a well-defined cardinality, which is a (possibly nonstandard) positive integer, as well as a well-defined parity, sum, etc. By a **hyperfinite set**, we mean a *-finite set whose cardinality is not a standard integer. We similarly talk about hyperfinite structures, orderings etc. to mean those whose cardinality is a nonstandard integer.

We can now talk formally about hyperfinite structures. However, we cannot assume in general that the semantic function for a given logic will agree with the semantics obtained by considering the structure "externally". However, for first-order logic, we have the following result:

Proposition 1. *[3] Let L be a language for which each symbol is internal, and let M be an L-structure such that the domain of M is internal and the interpretation of each symbol in L is internal. Let $\phi(\vec{x})$ be a formula of L that has standard finite cardinality (i.e. number of symbols). Then the internal satisfaction predicate ${}^*\!\models$ agrees with the external satisfaction predicate $\models$ on ϕ. That is, if $\vec{c}$ is a finite sequence of parameters from M, then $M \,{}^*\!\models \phi(\vec{c})$ iff $M \models \phi(\vec{c})$.*

Given the above proposition we will not distinguish the two kinds of satisfaction predicates when we are dealing with finitary first-order ϕ's.

The nonstandard technique is now based on the following simple proposition:

Proposition 2. *[3] The following are equivalent for any boolean query Q on finite structures:*

- *There are two hyperfinite $^* -$ structures that agree on every query in the first-order language L but disagree on $^*Q$.*
- *Q is not expressible in L.*

It is now easy to formalize the inexpressibility arguments mentioned in the beginning of the section. We can form, for example, a graph consisting of two hyperfinite chains (such a graph exists by transfer plus saturation), and show, by a single Ehrenfeucht game, that this graph , considered as an infinite structure, is elementarily equivalent to any graph consisting of a single hyperfinite cycle. The inexpressibility result now follows from Proposition 2.

3.2 Indiscernibles

Although the use of nonstandard models eliminates the need to construct countably many counterexample models, and relieves some of the combinatorial burden in a game argument, it may still be difficult to reason about elementary equivalence in an arbitrary hyperfinite structure.

To relieve the amount of analysis necessary in analyzing elementary equivalence, we will often want to restrict our attention to models whose algebraic structure is "as simple as possible". Indiscernibility is a method for capturing the intuition that the domain of our structures should have no unnecessary algebraic dependencies among its elements. This idea is implicit in many of the Ramsey-theoretic constructions used in Ehrenfeucht-Fraisse constructions [21],[26].

We now define indiscernibles formally. Let I be any ordered set. A sequence $B = \langle B_i \rangle_{i \in I}$, whose elements come from an infinite L-structure M is **indiscernible** if for every formula $\phi(\vec{x})$, ϕ is satisfied in M by either every increasing (in the order on I) subsequence of B or by no increasing subsequence of B. Indiscernibles are discussed at length in [6].

Within an indiscernible set, the logical structure of the model reduces to a simple ordering. Indiscernibles were used in algorithms for eliminating constraints from constraint queries in [3] [26] and [21].

An infinite set of indiscernibles need not exist in an arbitrary infinite structure. For example, considering the structure $M = \langle N, +, < \rangle$, we easily see that there can be no indiscernible set of size bigger than 1! However, it is easy to show, using saturation, that for any infinite structure, there is an infinite set of indiscernibles in the nonstandard extension $^*M$: this makes the use of indiscernibles particularly powerful in conjunction with nonstandard methods.

In this work, we will consider the use of indiscernibles in collapsing logics with counting quantifiers to first-order logics. The construction in the next section will give an example of the use of indiscernibles to reduce every formula in the language $L(Ct)$ to a first-order formula. Since several previous expressibility bounds on aggregates make use of some sort of "Count Elimination" [19], we hope to investigate this phenomenon in more generality in forthcoming work.

4 Nonstandard Models and Unary Counters

Consider the logics $L(Ct)$, $L(Tm)$, and $L(Ct, \mathbf{N})$ defined in Section 2. We are interested in investigating the relative expressive powers of these languages, using the techniques mentioned above. In particular, we wish to show that $L(Ct)$ cannot express important properties expressible in $L(Tm)$, and $L(Ct, \mathbf{N})$. Our plan, of course, will be to find two nonstandard models which satisfy the same sentences of $L(Ct)$ but do not satisfy the same sentences of $L(Tm)$ or of $L(Ct, \mathbf{N})$.

Our first result will show that the language of unary counters $L(Ct)$ cannot count the number of equivalence classes in an equivalence relation, while $L(Ct, \mathbf{N})$ can express this. We now fix a particular first-order language L^0.

Definition 3. Let the language L^0 have one unary predicate symbol S and one binary predicate symbol E. Let θ be the following sentence of $L^0(Tm)$:

$$count\{count\{y : E(x,y)\} : x = x\} = count\{z : S(z)\}$$

If E is an equivalence relation, θ says that the number of distinct sizes of equivalence classes of E is equal to the number of elements of S.

Definition 4. Let θ^+ be the following sentence of $L^0(Ct, \mathbf{N})$:

$$\forall i[\forall j(Bit(j,i) \Leftrightarrow \exists x \; count\{y : E(x,y)\} = j) \Rightarrow Setcard(i) = count\{z : S(z)\}],$$

where i, j are variables of sort $\mathbf{N}$, $Bit(j,i)$ is true exactly when the jth bit of the binary representation of i is set, and $Setcard(i)$ is the cardinality of the set coded by the binary representation of i.

Since our arithmetic permits arbitrary functions on the integers, the predicates Bit and $Setcard$ are certainly expressible. Note that θ^+ expresses the same property as θ, that is, for every finite model $\mathcal{C}$ for L^0, $\mathcal{C}$ satisfies θ if and only if it satisfies θ^+. The sentence θ^+ does not have nested counts.

We shall prove the following theorem, which shows that neither θ nor θ^+ is expressible in $L^0(Ct)$, so that both $L(Tm)$ and $L(Ct, \mathbf{N})$ are proper extensions of $L(Ct)$.

Theorem 5. *For every sentence φ of $L^0(Ct)$ there is a finite model $\mathcal{C}$ in which θ is not equivalent to φ: i.e., the unary counter language cannot express θ.*

As mentioned above, our plan will be to find two nonstandard models which satisfy the same sentences of $L^0(Ct)$ but do not satisfy the same sentences of $L^0(Tm)$ or of $L^0(Ct, \mathbf{N})$.

We let $\mathcal{N}$ be the structure with universe set $\mathbf{N}$ and a symbol for $+$, $\times$, and every relation and function in the set Λ. For each *finite model $\mathcal{A}$ for L, each term t and formula φ of $L(Tm)$ is interpreted in the natural way, using the functions and relations on the extension ${}^*\mathcal{N}$. Terms of sort $\mathbf{N}$ are interpreted as functions with values in ${}^*\mathbf{N}$.

Our goal is to prove the following.

Theorem 6. *There exist *finite models $\mathcal{A}$ and $\mathcal{B}$ which satisfy the same sentences of $L^0(Ct)$ such that θ is false in $\mathcal{A}$ but true in $\mathcal{B}$.*

Given this theorem, the inexpressibility of θ in $L^0(Ct)$ now follows from the general results of the previous section.

The proof of this theorem will give a canonical example of the use of indiscernibles and nonstandard universes together. We will first show that a special set of indiscernibles exists, and then make use of them to prove Theorem 6.

We use the usual notation for intervals in ${}^*\mathbf{N}$; for example, $(J, K] = \{x \in {}^*\mathbf{N} : J < x \text{ and } x \leq K\}$. By a *finite sequence in ${}^*\mathbf{N}$ we mean an an element of the star-image of the finite sequences in $\mathbf{N}$. By the transfer principle, each *finite sequence in ${}^*\mathbf{N}$ is a function $\vec{d} = \langle d_1, \ldots, d_H\rangle$ from the interval $(0, H]$ into ${}^*\mathbf{N}$ for some hyperinteger $H \in {}^*\mathbf{N}$.

Lemma 7. *There is a strictly increasing *finite sequence $\vec{d} = \langle d_1, \ldots, d_H\rangle$ in ${}^*\mathbf{N}$ such that $0 < d_1 < H$ and $\vec{d}$ is indiscernible in ${}^*\mathcal{N}$; that is, any two finite increasing subsequences of $\vec{d}$ satisfy the same first order formulas in ${}^*\mathcal{N}$.*
Proof. By the Paris-Harrington form of Ramsey's theorem, for every $e, r \in \mathbf{N}$ there exists $m \in \mathbf{N}$ such that for every partition $P : [m]^e \to r$ there is a subset $Q \subseteq m$ which is homogeneous for P and has size $|Q| \geq \min(Q)$. Then for each finite set φ of formulas in the language of $\mathcal{N}$ there is a strictly increasing finite sequence $\langle d_1, \ldots, d_h\rangle$ in $\mathbf{N}$ such that $0 < d_1 < h$ and $\vec{d}$ is indiscernible for φ. Using saturation once again on ${}^*\mathcal{N}$, it follows that there is a strictly increasing *finite sequence $\vec{d} = \langle d_1, \ldots, d_H\rangle$ in ${}^*\mathbf{N}$ such that $0 < d_1 < H$ and $\vec{d}$ is indiscernible in ${}^*\mathcal{N}$. $\square$

Hereafter we let $\vec{d} = \langle d_1, \ldots, d_H\rangle$ be as in Lemma 7, and put $d_0 = 0$.

Corollary 8. *H and d_1 are infinite, and $\frac{d_J}{d_{J-1}}$ is infinite for each $J \in (1, H]$ (i.e. for each standard integer n, $d_J > n \cdot d_{J-1}$) for each $J \in (1, H]$).*
The proofs are in the appendix.

We now define the *finite structures $\mathcal{A}$ and $\mathcal{B}$ for L^0.

Definition 9. Let $\mathcal{A} = \langle A, E, S\rangle$ where $A = (0, d_H]$, E is the equivalence relation on A with equivalence classes $(d_{J-1}, d_J], J \in (0, H]$, and $S = (0, d_1]$. Let $K = d_1$ and $\mathcal{B} = \langle B, F, S\rangle$ where $B = (0, d_K]$, $F = E \cap B \times B$, and $S = (0, d_1]$.

Lemma 10. *The sentence θ is false in $\mathcal{A}$ and true in $\mathcal{B}$.*

Lemma 11. *$\mathcal{A}$ and $\mathcal{B}$ satisfy the same sentences of $L^0(Ct)$.*

Lemma 10 is proved straightforwardly by insepection; the proof of Lemma 11 is quite involved, and can be found in the appendix.

5 Applications to weakest preconditions

We now outline a version of the previous results for a language L' whose concrete syntax is closer to existing database languages. The description we give below is based on the tuple relational calculus with range restriction [1], and on our analysis of the PRL constraint language [14] [15].

We have a signature $\{R_1, \ldots, R_n\}$, and for each $i \leq n$ a finite set of **attribute symbols** A_i. The **arity** of R_i is the cardinality of A_i. For each R_i we also fix a countably infinite set U_i. We now define our language L'.

We have variables of sort i for each $i \leq n$, and an integer sort. A formula will be built out of atomic formulas of the form:

- $x.a = y.b$ where x and y are variables of the same sort i, and a and b are symbols in A_i, or of the form
- $P(\tau_1, \ldots, \tau_n)$, where P is an integer predicate and the τ_j are terms of type integer.

Formulae are built up by the logical connectives and quantifications $\forall x \in R_i\ \phi(x)$, and $\exists x \in R_i\ \phi(x)$, where x has sort i.

Terms are built up via composition from atomic terms, which are of one of the forms:

1. $f(\vec{x})$, where f is a symbol for some function from tuples of integers to integers,and $\vec{x}$ is a tuple of variables of integer sort. The corresponding language $L'(Ct)$ adds the ability to form terms from formulas via the rule:
2. $Count\{x \in R_i : \phi(x)\}$, where x has sort i and ϕ is a formula.

The language $L'(Tm)$ further supplements this by permitting

3. $Count\{\tau(x) : x \in R_i \wedge \phi(x)\}$, where τ is a term.

A structure for L' consists of an assignment to each relational variable R_i of a finite collection of elements of the set of functions $U_i^{A_i}$ (i.e. tuples). Satisfaction relative to an assignment of variables is defined exactly as in the language $L(Ct)$.

Let our signature have two relations R and S, and let R have attributes a and b, while S has attributes c and d .

Theorem 12. *The $L'(Tm)$ sentence θ' below is not expressible in $L'(Ct)$:*

$$\theta' \equiv count\{x.a : x \in R \wedge x = x\} = count\{z \in S : z = z\}.$$

The proof is found in the appendix.

5.1 Preconditions and Definable transactions

Let $\mathcal{L}$ be any of the standard logical languages (first-order logic, infinitary logic, etc.). We let $s(\mathcal{L})$ denote the sentences of $\mathcal{L}$. We will talk about a database (that is, a finite structure as in the previous section) satisfying a sentence or open formula of $\mathcal{L}$: we mean this in the usual sense. By a **transaction** on databases, we mean simply any function mapping databases to databases. In the following discussion, we will let D range over databases for a particular signature, and $\mathcal{T}$ denote the set of database transactions for this signature. $TERM(\mathcal{L})$ denotes the set of terms of $\mathcal{L}$ and for Γ a collection of terms and D a database, we let $\Gamma(D)$ denote all elements obtained from applying a vector of terms in Γ to elements in the underlying set of D. That is:

$$\Gamma(D) = \{\vec{\tau}(\vec{y}) : \vec{y} \subset D \text{ and } \vec{\tau} \subset \Gamma\}$$

We now discuss two classes of transactions on finite models associated with $\mathcal{L}$. The definitions are taken from [2]. The class $\mathcal{WPC}(\mathcal{L})$ (transactions with *weakest preconditions* with respect to $\mathcal{L}$) is defined as

$$\{T \in \mathcal{T} \mid \exists \text{ recursive } f : s(\mathcal{L}) \to s(\mathcal{L}) :\ \forall D\ \forall \alpha \in s(\mathcal{L}) : T(D) \models \alpha \Leftrightarrow D \models f(\alpha)\}$$

A transaction T has weakest preconditions for $\mathcal{L}$ if we can statically determine whether the database resulting from T will satisfy a constraint in $\mathcal{L}$.

The set of *L-definable* transactions is the collection

$$\begin{aligned}\mathcal{DEF}(\mathcal{L}) = \{T \mid \forall R\ \exists\ \mathcal{L} \text{ formula } \beta_R(\vec{x})\ \exists\ \Gamma \subset TERM(\mathcal{L}) \text{ such that}\\ \forall D\ \forall \vec{t} : T(D) \models R(\vec{t}) \Leftrightarrow \vec{t} \subset \Gamma(D) \wedge\ D \models \beta_R(\vec{t})\}\end{aligned}$$

The class of definable transactions are those that can be expressed using a finite set of $\mathcal{L}$ terms and $\mathcal{L}$ formulae.

In [2] it is observed that for first-order languages, we have containment, $\mathcal{DEF}(\mathcal{L}) \subset \mathcal{WPC}(\mathcal{L})$. That is, definable transactions all admit weakest preconditions. This is a desirable closure property for a query language to possess. [2] investigates weakest preconditions over a number of logics, and over a number of transaction languages. In [15],[2], and [14] applications of weakest precondition closure to database integrity maintenance are discussed.

Given the importance of weakest precondition closure for database query languages, it is important to see if natural extensions of the relational calculus, such as $L'(Ct)$, posess this closure property. It follows from the main result of this paper that containment does *not* hold for the language $L'(Ct)$.

Corollary 13. *There are $L'(Ct)$-definable transactions that do not posess weakest preconditions over $L'(Ct)$.*

6 Conclusions and future work

Languages with aggregate constructs are not nearly so well understood as the relational calculus. In particular, issues of optimization, complexity, safety, and expressiveness remain open for many models of aggregation. We are interested in developing a usable set of rewrite-rules for simplifying languages such as $L(Ct)$ and $L(Tm)$, and getting semantic characterizations of the definable transactions that are available (along the lines of Gaifman's locality theorem [11] or the bounded degree property of [19]).

We are interested in studying the relationship of the language $L(Tm)$ to various other counting languages (those discussed, for example, in [16]). In particular, it would be helpful to know whether languages with binary counters can express all sentences of $L(Tm)$, and similarly for n-ary counters.

Techniques for proving expressiveness bounds on query languages are hard to come by. We've presented here one technique for analyzing expressivity of query languages, based on the use of indiscernibles and nonstandard models, that we believe can be useful outside of the context of aggregates. In particular, we hope to investigate the interaction of these techniques with the use of Ehrenfeucht games for logics with counting [16] [12].

Questions in this work were motivated by considering the closure under weakest preconditions of various extensions of first-order logic. The closure of a specification language under weakest-preconditions of definable transactions is helpful for integrity constraint maintenance [2],[22],[23]. It is therefore important to find logics that include aggregate operators that have this closure property, are of manageable complexity, and allow for optimization and analysis. The results of this paper show that $L(Ct)$ is not closed under weakest-preconditions, and we suspect that the same is true for $L(Tm)$. We would like to discover natural weakest-precondition closed logics containing these languages.

Acknowledgements: The authors wish to thank the referees for several helpful comments, and Leonid Libkin for numerous improvements to the text.

References

1. S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. M. Benedikt, T. Griffin, and L. Libkin Verifiable properties of database transactions. In *Proceedings of 15th ACM Symposium on Principles of Database Systems*, pages 117–128, Montreal Canada, June 1996.
3. M. Benedikt, G. Dong, L. Libkin, L. Wong. Relational expressive power of constraint query languages. In *Proceedings of 15th ACM Symposium on Principles of Database Systems*, pages 5–17, Montreal Canada, June 1996.
4. M. Benedikt and L. Libkin. On the structure of queries in constraint query languages. In *Proceedings of 11th IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey* 1996.
5. P. Buneman, S. Naqvi, V. Tannen, L. Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149(1):3–48, 1995.
6. C. C. Chang and H. Jerome Keisler. *Model Theory.* North-Holland Elsevier 1990.
7. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer Verlag, 1995.
8. A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamentae Mathematicae*, 49:129–141, 1961.
9. K. Etessami. Counting quantifiers, successor relations, and logarithmic space. In *Proceedings of 10th IEEE Conference on Structure in Complexity Theory*, May 1995, pages 2–11.
10. K. Etassami and N. Immerman. Tree canonization and transitive closure. Tenth Annual IEEE Symposium on Logic in Computer Science. 1995.
11. H. Gaifman. On local and nonlocal properties. In J. Stern, editor, *Logic Colloquium '81*, pages 105–135. North -Holland, 1982.

12. Gradel, E. and Otto, M. Inductive definability with counting on finite structures. In E. Borger (ed.), *Computer Science Logic*, LNCS 702 Springer (1993), 231-247.
13. Erich Gradel and Yuri Gurevich. Metafinite model theory. In *Logic and Computational Complexity, International Workshop LCC'94* Indianapolis, IN,313-366.
14. T. Griffin and H. Trickey. Integrity Maintenance in a Telecommunications Switch. In IEEE Data Engineering Bulletin V. 17, No. 2., June 1994.
15. T. Griffin, H. Trickey, and C. Tuckey. Update constraints for relational databases. Technical Memorandum AT&T Bell Laboratories, 1992
16. S. Grumbach and C. Tollu. On the expressive power of counting. Fourth International Conference on Database Theory. 1992.
17. N. Immerman, E.S. Lander. Describing graphs: a first-order approach to graph canonization. In *Complexity Theory Retrospective*, pp. 59-81. Springer, 1990.
18. C. Karp. Finite quantifier equivalence. In *The Theory of Models*, edited by J. Addison, L. Henkin, and A. Tarski, North-Holland 1965, 407-412.
19. L. Libkin and L. Wong. New techniques for studying set languages, bag languages and aggregate functions. In *Proceedings of the 13th Conference on Principles of Database Systems,* Minneapolis MN, May 1994, pages 155–166.
20. L. Libkin and L. Wong. On representation and querying incomplete information in databases with bags. Information Processing Letters 56 (1995), 209-214
21. J. Paredaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over the reals. In *Proceedings of 10th IEEE Symposium on Logic in Computer Science, San Diego, California*, pages 79–87, 1995.
22. X. Qian. An effective method for integrity constraint simplification. In *Fourth International Conference on Data Engineering*, 1988.
23. X. Qian. *The Deductive Synthesis of Database Transactions.* PhD thesis, Stanford University, 1989.
24. O.Belagradek, A. Stolboushkin, M. Tsaitlin. On order-generic queries. Manuscript. To appear.
25. V. Tannen, Tutorial: Languages for collection types, *in* "Proceedings of 13th Symposium on Principles of Database Systems," Minneapolis, May 1994.
26. J. Van Den Bussche and M. Otto. First-order queries on databases embedded in an infinite. structure. Information Processing Letters, to appear.

A Proofs

We will prove the remaining lemmas used to prove Theorem 6. Hereafter we let $\vec{d} = \langle d_1, \ldots, d_H \rangle$ be as in Lemma 7, and put $d_0 = 0$.

Proof of corollary 8. By indiscernibility, d_1 is infinite, and thus H is infinite. Since $\vec{d}$ is strictly increasing and $d_1 < H$, we have $2d_1 \leq d_H$. By indiscernibility, $2d_{J-1} \leq d_J$ for each $J \in (1, H]$, and by indiscernibility again, $2^n d_{J-1} \leq d_J$ for each J and each finite n. $\Box$

Proof of lemma 11. We have $count\{x : S(x)\} = d_1$ in both $\mathcal{A}$ and $\mathcal{B}$. In $\mathcal{A}$, $count\{count\{y : E(x,y)\} : x = x\} = H > d_1$, so θ fails in $\mathcal{A}$. In $\mathcal{B}$, $count\{count\{y : E(x,y)\} : x = x\} = d_1$, so θ holds in $\mathcal{B}$. $\Box$

It remains to show that $\mathcal{A}$ and $\mathcal{B}$ satisfy the same sentences of $L^0(Ct)$. In order to do this we introduce an auxiliary first order vocabulary L^1 and corresponding models $\mathcal{A}^1$ and $\mathcal{B}^1$.

Definition 14. Let $f : A \to (0, H]$ be the function such that $f(x) = J$ whenever $J \in (0, H]$ and $x \in (d_{J-1}, d_J]$. Let L^1 be a first order vocabulary with countably many unary relations $\min_n(x), \max_n(x)$ and countably many binary relations $x \preceq_n y$ for $n \in \mathbf{N}$. Let $\mathcal{A}^1$ be the model for L^1 with universe A such that

- $\mathcal{A}^1 \models \min_n(x)$ iff $f(x) \leq n$,
- $\mathcal{A}^1 \models \max_n(x)$ iff $H - n \leq f(x)$,
- $\mathcal{A}^1 \models x \preceq_n y$ iff $f(x) + n \leq f(y)$.

Let $\mathcal{B}^1$ be the model for L^1 with universe B such that

- $\mathcal{B}^1 \models \min_n(x)$ iff $f(x) \leq n$,
- $\mathcal{B}^1 \models \max_n(x)$ iff $K - n \leq f(x)$,
- $\mathcal{B}^1 \models x \preceq_n y$ iff $f(x) + n \leq f(y)$.

Lemma 15. *For every formula $\varphi(\vec{x})$ of $L^0(Ct)$ there is a formula $\varphi^1(\vec{x})$ of $L^1(Ct)$ such that for all $\vec{a}$ in A, $\mathcal{A} \models \varphi[\vec{a}]$ if and only if $\mathcal{A}^1 \models \varphi^1[\vec{a}]$, and for all $\vec{b}$ in B, $\mathcal{B} \models \varphi[\vec{b}]$ if and only if $\mathcal{B}^1 \models \varphi^1[\vec{b}]$.*
Proof: Put $(x = y)^1 = (x = y)$, $E(x, y)^1 = (x \preceq_0 y \wedge y \preceq_0 x)$, and $S(x)^1 = min_1(x)$. Then use the formation rules in the obvious way to define φ^1 for arbitrary φ. □

Definition 16. Let $\vec{a} \equiv_0 \vec{b}$ mean that $|\vec{a}| = |\vec{b}|$ and $(\mathcal{A}^1, \vec{a})$ and $(\mathcal{B}^1, \vec{b})$ satisfy the same atomic formulas of L^1. We use a similar notation for pairs of tuples which are both in A or in B.

Lemma 17. *$\mathcal{A}^1$ and $\mathcal{B}^1$ satisfy the same sentences of L^1. In fact, $\vec{a} \equiv_0 \vec{b}$ if and only if $(\mathcal{A}^1, \vec{a})$ and $(\mathcal{B}^1, \vec{b})$ satisfy the same formulas of L^1.*
Proof: Using the fact that we have $\preceq_n$ as an atomic relation in the language for each n, and the fact that $\mathcal{A}^1$ and $\mathcal{B}^1$ are countably saturated, we derive that the relation $\vec{a} \equiv_0 \vec{b}$ has the back and forth property. It follows that whenever $\vec{a} \equiv_0 \vec{b}$, player $\exists$ has a winning strategy in the Ehrenfeucht-Fraisse game with ω moves between $(\mathcal{A}^1, \vec{a})$ and $(\mathcal{B}^1, \vec{b})$. In addition, we have that the empty sequences from each model are $\equiv_0$-equivalent, since there are no atomic sentences in L^1. Thus $(\mathcal{A}^1, \vec{a})$ and $(\mathcal{B}^1, \vec{b})$ are elementarily equivalent by Karp's theorem in [18]. □

Lemma 18. *Let $\Gamma(\vec{x})$ be a set of quantifier-free formulas of L^1 maximal consistent with the theory of $\mathcal{A}^1$ and let $\psi(\vec{x}, y)$ be a quantifier-free formula in L^1. Let $s = domain(\vec{x}) \cup \{\min, \max\}$. There exist $\alpha_1, \ldots, \alpha_m \in s, \beta_1, \ldots, \beta_m \in \{-1, 1\}, \gamma_1, \ldots, \gamma_m \in \mathbf{Z}, \delta \in \mathbf{Z}$ such that whenever $(\mathcal{A}^1, \vec{a})$ satisfies $\Gamma(\vec{x})$,*

$$count\{y : \psi(\vec{a}, y)\} = \delta + \sum_{j=1}^{m} \beta_j d_{f(a_{\alpha_j}) + \gamma_j},$$

and whenever $(\mathcal{B}^1, \vec{b})$ satisfies $\Gamma(\vec{x})$,

$$count\{y : \psi(\vec{b}, y)\} = \delta + \sum_{j=1}^{m} \beta_j d_{f(b_{\alpha_j}) + \gamma_j},$$

with the convention that $f(a_{\min}) = f(b_{\min}) = 0$, $f(a_{\max}) = H$, *and* $f(b_{\max}) = K$.
Proof: Given $\Gamma(\vec{x})$, any quantifier-free formula $\psi(\vec{x}, y)$ of L^1 says that either y belongs to a subset of $\{x_i : i < |\vec{x}|\}$, or $y \notin \{x_i : i < |\vec{x}|\}$ and $f(y)$ belongs to a finite union of disjoint "intervals" with endpoints at a finite distance from elements of

$$\{f(x_i) : i \in s\}.$$

In the model $(\mathcal{A}, \vec{a})$, such an interval will have the form $(u + \gamma, u' + \gamma']$ where $\gamma, \gamma' \in \mathbf{Z}$ and u, u' belong to $\{f(x_i) : i \in s\}$. The number of elements y of A such that $f(y)$ belongs to such an interval is equal to the difference $d_{u'+\gamma'} - d_{u+\gamma}$. A similar computation holds for $(\mathcal{B}, \vec{b})$. □

We now prove our main lemma, which shows that in the models $\mathcal{A}^1$ and $\mathcal{B}^1$ both quantifiers and counts can be eliminated.

Lemma 19. *For each formula* φ *of* $L^1(Ct)$ *there is a quantifier-free formula* φ^1 *of* L^1 *such that* $\mathcal{A}^1 \models \varphi \Leftrightarrow \varphi^1$ *and* $\mathcal{B}^1 \models \varphi \Leftrightarrow \varphi^1$.
Proof: We argue by induction on the complexity of φ. As remarked earlier, we may assume without loss of generality that φ has no quantifiers, because the existential quantifier $\exists x \psi(x, \ldots)$ may be replaced by $\neg count\{x : \psi(x, \ldots)\} = 0$. The hard case of the induction is the case where φ has the form

$$r(count\{y : \psi_i(\vec{x}, y)\} : i \leq j)$$

for some j-ary relation r on $\mathbf{N}$. For simplicity we let $j = 1$, so that

$$\varphi = r(count\{y : \psi(\vec{x}, y)\}).$$

By inductive hypothesis we have a quantifier-free formula $\psi^1(\vec{x}, y)$ of L^1 which is equivalent to ψ in both models.
Claim 1. Suppose $\vec{a} \equiv_0 \vec{b}$. Then $(\mathcal{A}^1, \vec{a}) \models \varphi$ if and only if $(\mathcal{B}^1, \vec{b}) \models \varphi$.
Proof of Claim 1: By the preceding lemma, $(\mathcal{A}^1, \vec{a})$ satisfies equation (18) and $(\mathcal{B}^1, \vec{b})$ satisfies the corresponding equation (18). The claim now follows by the indiscernibility of the sequence $\vec{d}$ in ${}^*\mathcal{N}$.
Claim 2. There is a quantifier-free formula $\varphi^1(\vec{x})$ of L^1 such that $\mathcal{A}^1 \models \varphi \Leftrightarrow \varphi^1$.
Proof of Claim 2: Let $\Sigma(\vec{x})$ be the set of all quantifier-free formulas $\sigma(\vec{x})$ of L^1 such that $\mathcal{A}^1 \models \varphi \Rightarrow \sigma$. Suppose $\mathcal{A}^1 \models \Sigma[\vec{a}]$. Then the set of formulas

$$\{\varphi(\vec{x})\} \cup \{\eta(\vec{x}) : \eta \text{ is quantifier-free in } L^1 \text{ and } \mathcal{A}^1 \models \eta[\vec{a}]\}$$

is finitely satisfiable in $\mathcal{A}^1$. Since ${}^*\mathcal{N}$ is saturated, this set of formulas is satisfiable in $\mathcal{A}^1$ by some tuple $\vec{c}$. Then $\mathcal{A}^1 \models \varphi[\vec{c}]$ and $\vec{a} \equiv_0 \vec{c}$. By Lemma 17 and the saturation of ${}^*\mathcal{N}$, there exists $\vec{b}$ in B such that $\vec{a} \equiv_0 \vec{b} \equiv_0 \vec{c}$. By Claim 1, $(\mathcal{B}^1, \vec{b}) \models \varphi$ and $(\mathcal{A}^1, \vec{a}) \models \varphi$. Thus every tuple which satisfies Σ in $\mathcal{A}^1$ satisfies φ. Since ${}^*\mathcal{N}$ is saturated, there is a finite conjunction $\varphi^1(\vec{x})$ of formulas in $\Sigma(\vec{x})$ such that $\mathcal{A}^1 \models \varphi \Leftrightarrow \varphi^1$, and the claim is proved.

Now let $\vec{b}$ be a tuple in B. By Lemma 17 and the saturation of ${}^*\mathcal{N}$, there exists $\vec{a}$ in A such that $\vec{a} \equiv_0 \vec{b}$. By Claim 2, $\mathcal{A}^1 \models (\varphi \Leftrightarrow \varphi^1)[\vec{a}]$. By Claim 1, $\mathcal{B}^1 \models (\varphi \Leftrightarrow \varphi^1)[\vec{b}]$. □

Corollary 20. *$\mathcal{A}^1$ and $\mathcal{B}^1$ satisfy the same sentences of $L^1(Ct)$. Moreover, $\mathcal{A}$ and $\mathcal{B}$ satisfy the same sentences of $L^0(Ct)$.*

This completes the proof of Lemma 11, and hence Theorem 6.

Proof of Theorem 12. Fix some bijection K from the universe for tuples of S to the universe for tuples of R. Let $\mathcal{M}$ be the class of L' models of the form $M' = \langle R, S \rangle$ which satisfy

- the sets $\{x.a : x \in R\}$, $\{x.b : x \in R\}$, $\{x.c : x \in S\}$, are pairwise disjoint.
- $x.c = x.d$ for each $x \in S$, $\quad - \; x.c \neq y.c$ for distinct x, y in S.
- For each x in S, $K(x.c)$ is in R.

We define a mapping F that maps $\mathcal{M}$ to models for the language L^0 defined previously. $F(M')$ is defined to have domain equal to R, the predicate S is interpreted by the K-image of the attributes of the relation S in M' , and the interpretation of the binary predicate E is defined by $xEy \leftrightarrow x.a = y.a$.

We also define a mapping G from formulae and terms of $L(Ct)$ to formulae and terms (respectively) of $L'(Ct)$ as follows:
$G(x.a = y.a) = xEy$, where x and y are any two variables of sort R, (not necessarily distinct).
$G(x.c = y.c) = G(x.d = y.d) = G(x.c = y.d) = G(x.d = y.c) = x = y$, where x, y have sort S.
$G(x.att1 = y.att2) = false$, for all attribute/sort combinations not listed above.
$G(P(\tau_1, \ldots, \tau_n)) = P(G(\tau_1), \ldots, G(\tau_n)), \quad G(\forall x \in R \; \phi(x)) = \forall x \; G(\phi(x))$
$G(\exists x \in R \; \phi(x)) = \exists x \; G(\phi(x)), \quad G(\forall x \in S \; \phi(x)) = \forall x \; S(x) \wedge \; G(\phi(x))$
$G(\exists x \in S \; \phi(x)) = \exists x \; S(x) \wedge \; G(\phi(x)), \quad G(f(\vec{x})) = f(\vec{x}),$
$G(Count\{x \in R : \phi(x)\}) = Count\{x : G(\phi(x))\}$.

Proposition 21. *1) The maps F and G are surjections. 2) $M' \models \phi(\vec{x}) \Leftrightarrow F(M') \models G(\phi(\vec{x}))$.*
1) is proved by inspection and 2) by straightforward induction on complexity.

Theorem 12 now follows, since if there were a formula of $L'(Ct)$ expressing exactly those models that satisfy θ', then by applying the inverse of G to this formula we would get a sentence of $L(Ct)$ expressing exactly those models satisfying θ , contradicting the main theorem of the previous section.

Proof of Corollary 13. Consider the transaction T on structures for the signature SC defined above as follows: $R \Leftarrow \Pi_a(R)$, $S \Leftarrow S$, where $\Pi_a(R)$ is defined via $x \in \Pi_a(R) \Leftrightarrow (\exists \; y \in R \; x.a = y.a \; \wedge \; x.b = y.a)$. Then T is clearly $L(Ct)$-definable. The precondition of the $L'(Ct)$-sentence $count\{y \in R : y = y\} = count\{z \in S : z = z\}$ is exactly the sentence θ' of Theorem 12 cited above, which is not expressible in $L'(Ct)$.

This article was processed using the LaTeX macro package with LLNCS style