Converting untyped formulas to typed ones

Jan Van den Bussche Luca Cabibbo

Abstract

We observe that every first-order logic formula over the untyped version of some many-sorted vocabulary is equivalent to a union of many-sorted formulas over that vocabulary. This result has as direct corollary a theorem by Hull and Su on the expressive power of active-domain quantification in the relational calculus.

Keywords: Many-sorted logic, relational database, relational calculus

1 Introduction

Many-sorted logic is widely used in the formal aspects of computer science as a "typed" version of classical (first-order) logic which is in principle "untyped", or, as one also calls it, *one-sorted*. In this short paper, we compare many-sorted and one-sorted first-order logic with respect to their expressive power. It is well-known that many-sorted logic can be simulated using one-sorted logic. We prove a converse to this simulation.

Specifically, suppose we work in some fixed many-sorted vocabulary σ. There are two basic ways to express first-order logic properties of σ-structures. One can use standard many-sorted σ-formulas, which are well-typed in the sense that each variable occurring in them is restricted to range only over elements of a specified sort. Or one can use untyped formulas, where variables can range over elements of all sorts, and in which predicates are available to test whether an element belongs to a particular sort. It is well known that every well-typed formula is equivalent to an untyped formula. We show that conversely, every untyped formula is equivalent to a union of well-typed formulas. (The union appears simply because a single untyped formula can be true for differently-sorted valuations of its free variables, which is impossible for well-typed formulas by their very nature.)

As an application, we show how this observation yields as a direct corollary an important theorem by Hull and Su on the expressive power of active-domain quantification in the relational calculus as a query language for relational databases. We thus obtain a simple proof of this theorem, which was originally proven in a rather complicated way.

The reader is supposed to be familiar with elementary mathematical logic.

2 Definitions

We start by defining vocabularies and structures over a vocabulary.

Definition 1 A *(relational) vocabulary* is a triple $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, where

1. σ_1 is a finite set of symbols called *sort symbols*;
2. σ_2 is a set of symbols called *relation symbols*; and
3. σ_3 is a mapping on the set of relation symbols, assigning to each relation symbol R a tuple $\sigma_3(R)$ of sort symbols.

The tuple $\sigma_3(R)$ is called *the type of R in σ*. Generally, a tuple of sort symbols is called *a type*.

Without ambiguity we will drop the subscript 3 and henceforth write $\sigma(R)$ instead of $\sigma_3(R)$.

Definition 2 A *structure* over a vocabulary σ is a pair $A = (A_1, A_2)$, where

1. A_1 is a function mapping each sort symbol s of σ to a set $A_1(s)$, such that different sort names get disjoint sets; and
2. A_2 is a function mapping each relation symbol R of σ to a subset $A_3(R)$ of $A_1(s_1) \times \cdots \times A_1(s_n)$, if $\sigma(R) = (s_1, \ldots, s_n)$.

Generally, any subset of $A_1(s_1) \times \cdots \times A_1(s_n)$ is called a *relation of type $(s_1, \ldots, s_n)$ on A* and its elements are called *tuples of type $(s_1, \ldots, s_n)$ on A*. The elements of the set $A_1(s)$, for a sort symbol s, are called *the elements of sort s in A*.

Without ambiguity we will drop the subscripts 1 and 2 and henceforth write $A(s)$ instead of $A_1(s)$ and $A(R)$ instead of $A_2(R)$.

If a vocabulary contains only one sort symbol then we call it *one-sorted*. We can associate a one-sorted vocabulary to each vocabulary as follows:

Definition 3 Let σ be a vocabulary. The *untyped vocabulary associated with σ*, denoted by $\overline{\sigma}$, is the vocabulary defined as follows:

1. There is only one sort symbol, 1;
2. The relation symbols of $\overline{\sigma}$ are the relation symbols of σ plus the sort symbols of σ;
3. For each sort symbol s of σ, $\overline{\sigma}(s) := (1)$, and for each relation symbol R of σ, $\overline{\sigma}(R) := (1, \ldots, 1)$, n times, where n is the length of the tuple $\sigma(R)$.

A structure over a one-sorted vocabulary is also called *one-sorted*. We can view every structure as a one-sorted structure in the following way:

Definition 4 Let A be a structure over σ. The *untyped structure associated with* A, denoted by $\overline{A}$, is the structure over $\overline{\sigma}$ defined as follows. We have $\overline{A}(1) := \bigcup_s A(s)$, where the union ranges over all sort symbols s of σ. Furthermore, we have $\overline{A}(s) := A(s)$ for each sort symbol s of σ and $\overline{A}(R) := A(R)$ for each relation symbol R of σ.

Now fix a vocabulary σ. For each sort symbol s of σ, we assume given an unbounded supply of *variables of sort* s. A *formula over* σ is a standard first-order logic formula φ in the language of equality and the relation symbols of σ, such that all variables occuring in φ are of a sort of σ.

Let A be a structure over σ and let φ be a formula over σ with free variables $x_1, \ldots, x_n$. Let the sort of x_i be s_i, and let $a_i \in A(s_i)$ for each $i = 1, \ldots, n$. By $A \models \varphi[a_1, \ldots, a_n]$, we denote that φ is true in A under the substitution of a_i for x_i for $i = 1, \ldots, n$. The notion of truth is the standard one, with the important provision that for each quantified variable x in φ, if the sort of x is s then x ranges only over the elements of sort s in A. The *relation defined by* φ *on* A, denoted by $A(\varphi)$, is then defined as

$$\{(a_1, \ldots, a_n) \mid a_i \in A(s_i) \text{ for } i = 1, \ldots, n \text{ and } A \models \varphi[a_1, \ldots, a_n]\}.$$

Note that this is a relation of type $(s_1, \ldots, s_n)$ on A.

3 From untyped to well-typed

We are now ready to prove the following basic lemma. The reason why it holds is the condition on structures imposed in Definition 2 that sets of elements of different sorts must be disjoint.

Lemma 1 *Let σ be a vocabulary. For every formula ψ over the associated untyped vocabulary $\overline{\sigma}$ there exist a finite number of formulas $\varphi_1, \ldots, \varphi_k$ over σ such that for each structure A over σ,*

$$A(\psi) = A(\varphi_1) \cup \cdots \cup A(\varphi_k).$$

The above equation will also be written in short as $\psi \equiv \varphi_1 \cup \cdots \cup \varphi_k$.

Proof. By induction on ψ.

- ψ is $x = y$. Then k is the number of sort symbols of σ, and φ_i is $x_i = y_i$, where x_i and y_i are variables of the ith sort of σ, for $i = 1, \ldots, k$.
- ψ is $s(x)$, with s a sort symbol. Then $k = 1$ and φ_1 is $y = y$ with y a variable of sort s.
- ψ is $R(x_1, \ldots, x_n)$, with R a relation symbol. Then $k = 1$ and φ_1 is $R(y_1, \ldots, y_n)$, where y_i is a variable of sort s_i, for $i = 1, \ldots, k$, if $\sigma(R) = (s_1, \ldots, s_n)$.

- ψ is $(\chi \vee \theta)$. By induction, we have $\chi \equiv \varphi_1 \cup \cdots \cup \varphi_l$ and $\theta \equiv \vartheta_1 \cup \cdots \cup \vartheta_m$. Let the number of variables occurring free in χ but not in θ be k, and let the number of variables occurring free in θ but not in χ be n. Then

$$\psi \equiv \bigcup_{\vec{t}} (\varphi_1 \wedge \phi_{\vec{t}}) \cup \cdots \cup \bigcup_{\vec{t}} (\varphi_l \wedge \phi_{\vec{t}}) \cup \bigcup_{\vec{t'}} (\vartheta_1 \wedge \phi_{\vec{t'}}) \cup \cdots \cup \bigcup_{\vec{t'}} (\vartheta_m \wedge \phi_{\vec{t'}}),$$

  where $\vec{t}$ ranges over all sorts of length n, $\vec{t'}$ ranges over all sorts of length k, and $\phi_{\vec{t}}$, for a sort $\vec{t}$, is the trivial formula defining the full relation of sort $\vec{t}$.[1] The variables used in the formulas $\phi_{\vec{t}}$ and $\phi_{\vec{t'}}$ must not occur in the formulas φ_i and ϑ_j.

- ψ is $(\exists x)\chi$. By induction, we have $\chi \equiv \varphi_1 \cup \cdots \cup \varphi_k$. We distinguish two possibilities:

  - x does not occur free in χ. Then ψ is equivalent to χ and thus $\psi \equiv \varphi_1 \cup \cdots \cup \varphi_k$.
  - x occurs free in χ. If ψ has n free variables, then χ has $n+1$, and we may assume that x is the $n+1$th. Now let y_i be the $n+1$th free variable of φ_i, for $i = 1, \ldots, k$. Then $\psi \equiv (\exists y_1)\varphi_1 \cup \cdots \cup (\exists y_k)\varphi_k$.

- ψ is $(\neg\chi)$. This is the only case that is not entirely trivial. By induction, we have $\chi \equiv \varphi_1 \cup \cdots \cup \varphi_l$. For every structure A, $\overline{A}(\psi)$ is the complement of $\overline{A}(\chi)$. Assume ψ has n free variables. Let $\vec{s_1}, \ldots, \vec{s_k}$ be an enumeration of all sorts of length n over σ. We may assume that φ_i defines a relation of sort $\vec{s_i}$, for $i = 1, \ldots, l$ (note that $l \leq k$). Since the n-tuples on A can be partitioned according to their sort, the complement of $\overline{A}(\chi)$ consists of the union over $i = 1, \ldots, k$ of the complements of $A(\varphi_i)$ *taken within the class of tuples of sort* $\vec{s_i}$, together with all tuples of some sort $\vec{s_j}$ with $j > l$. Hence, we have $\psi \equiv (\neg\varphi_1) \cup \cdots \cup (\neg\varphi_l) \cup \varphi_{l+1} \cup \cdots \cup \varphi_k$, where for $j > l$ the formula φ_j is the trivial formula defining the full relation of sort $\vec{s_j}$.

This completes the proof of the lemma.

As an immediate corollary, we obtain:

Proposition 1 *Let σ be a vocabulary, and let ψ be a formula over the associated untyped vocabulary $\overline{\sigma}$. Assume ψ has n free variables, and let $(s_1, \ldots, s_n)$ be a type over σ. Then there exists a formula φ over σ such that for each structure A over σ, the relation*

$$\{(a_1, \ldots, a_n) \mid a_i \in A(s_i) \text{ for } i = 1, \ldots, n \text{ and } \overline{A} \models \psi[a_1, \ldots, a_n]\} \qquad (*)$$

equals $A(\varphi)$.

[1] The trivial formula defining the full relation of sort $(s_1, \ldots, s_n)$ is $(x_1 = x_1) \wedge \cdots \wedge (x_n = x_n)$, where x_i is a variable of sort s_i for $i = 1, \ldots, n$.

Proof. Let $x_1, \ldots, x_n$ be the free variables of ψ, and consider the formula

$$\chi := \psi \wedge s_1(x_1) \wedge \cdots \wedge s_n(x_n).$$

Then $\overline{A}(\chi)$ is precisely the relation $(*)$. Applying the lemma to χ yields formulas $\varphi_1, \ldots, \varphi_k$ over σ such that $\overline{A}(\chi) = A(\varphi_1) \cup \cdots \cup A(\varphi_k)$. Since $\overline{A}(\chi)$ is a relation of type $(s_1, \ldots, s_n)$, the same is true for each $A(\varphi_i)$ and hence we can put $\varphi := \varphi_1 \vee \cdots \vee \varphi_k$, where we identify the free variables of the different φ_i. Thus the proposition holds.

Note that by specializing the proposition to sentences (formulas without free variables), we get that for each one-sorted sentence ψ there exists a many-sorted sentence φ such that $\overline{A} \models \psi$ iff $A \models \varphi$ for each A.

Example 1 As an illustration, assume σ has as sort symbols 1, 2 and 3, and as relation symbols R and S with $\sigma(R) = (1, 2)$ and $\sigma(S) = (1, 3)$. Consider the untyped formula $\psi(x) = (\exists y)(R(x, y) \vee S(x, y))$. This formula defines a relation of sort (1). To find a well-typed formula equivalent to ψ we first proceed according to the lemma. We have $R(x, y) \equiv R(x_1, x_2)$, where x_1 is of sort 1 and x_2 of sort 2. Similarly, $S(x, y) \equiv S(z_1, z_3)$, where z_1 is of sort 1 and z_3 of sort 3. We thus have $\psi \equiv (\exists x_2)R(x_1, x_2) \cup (\exists z_3)S(z_1, z_3)$. We can now identify the free variables x_1 and z_1 as in the proof of the proposition and obtain the well-typed formula $\varphi(x) = (\exists x_2)R(x, x_2) \vee (\exists z_3)S(x, z_3)$ equivalent to ψ.

Now consider the untyped sentence $\chi = (\forall x)(1(x) \vee 2(x))$. Then for any structure A over σ, $\overline{A} \models \chi$ iff $3^A = \emptyset$ (i.e., there are no elements of sort 3 in A). To find a many-sorted sentence equivalent to χ we again proceed first as in the lemma. We have $1(x) \equiv (x_1 = x_1)$ and $2(x) \equiv (x_2 = x_2)$, with x_1 of sort 1 and x_2 of sort 2. Rewriting χ as $\neg(\exists x)\neg(1(x) \vee 2(x))$, we further have $\neg(1(x) \vee 2(x)) \equiv \neg(x_1 = x_1) \cup \neg(x_2 = x_2) \cup (x_3 = x_3)$, with x_3 of sort 3, and thus $(\exists x)\neg(1(x) \vee 2(x)) \equiv (\exists x_1)\neg(x_1 = x_2) \cup (\exists x_2)\neg(x_2 = x_2) \cup (\exists x_3)(x_3 = x_3)$. Since there are no longer free variables, we can replace each $\cup$ by $\vee$ and apply the final negation to the disjunction thus obtained:

$$\chi \equiv \neg((\exists x_1)\neg(x_1 = x_1) \vee (\exists x_2)\neg(x_2 = x_2) \vee (\exists x_3)(x_3 = x_3)),$$

or equivalently

$$\chi \equiv (\forall x_1)(x_1 = x_1) \wedge (\forall x_2)(x_2 = x_2) \wedge \neg(\exists x_3)(x_3 = x_3).$$

The first two factors in the above conjunction are trivial and the third one indeed expresses that there are no elements of sort 3.

4 Active-domain quantification

Let L be a finite relational first-order language, i.e., a finite set of relation symbols where each relation symbol has an associated arity. Let U be some fixed infinite universe of data elements. A *(relational) database over L* is an L-structure, in the standard sense of mathematical logic, the domain of which equals U and all of whose relations are finite. The *active domain* of a database D is the (finite) set of domain elements that actually appear in one of the relations of D.

Let $\psi(x_1, \ldots, x_n)$ be a first-order formula (with equality) over L. If $d_1, \ldots, d_n$ are elements of U we denote the truth of ψ on D with d_i substituted for x_i, for $i = 1, \ldots, n$, by $D \models \psi[d_1, \ldots, d_n]$ as usual. With the notation $D \models_{\text{adom}} \psi[d_1, \ldots, d_n]$, we mean that $\psi[d_1, \ldots, d_n]$ evaluates to true on D when we let the quantifiers in ψ range over the active domain of D only, rather than over the whole of U.

Hull and Su[2] showed that active-domain quantification is equally powerful as normal quantification:

Theorem 1 *For each $\psi(x_1, \ldots, x_n)$ there exists φ such that for each database D and elements $d_1, \ldots, d_n$ of the active domain of D,*

$$D \models \psi[d_1, \ldots, d_n] \quad \Longleftrightarrow \quad D \models_{\text{adom}} \varphi[d_1, \ldots, d_n].$$

We now show that this theorem has a simple proof assuming Proposition 1. Let σ be the vocabulary having as sort symbols 1 and 2, and as relation symbols those of L, with $\sigma(R) = (1, \ldots, 1)$, n times, where n is the arity of R in L. We can view each database D as a structure over σ, where 1^D equals the active domain of D and 2^D equals its complement (taken in U). Given ψ, we construct a one-sorted formula $\overline{\psi}$ over $\overline{\sigma}$ inductively as follows:

- If ψ is atomic then $\overline{\psi} := \psi$;
- If ψ is $(\neg\chi)$ or $(\chi \vee \theta)$ then $\overline{\psi}$ is $(\neg\overline{\chi})$ or $(\overline{\chi} \vee \overline{\theta})$ respectively;
- If ψ is $(\exists x)\chi$ then $\overline{\psi}$ is $(\exists x)(1(x) \wedge \chi) \vee (\exists x)(2(x) \wedge \chi)$.

Clearly, $D \models \psi[d_1, \ldots, d_n]$ iff $\overline{D} \models \overline{\psi}[d_1, \ldots, d_n]$. By Proposition 1, there exists a formula ϕ over σ such that $\overline{D} \models \overline{\psi}[d_1, \ldots, d_n]$ iff $D \models \phi[d_1, \ldots, d_n]$. By Lemma 2 (stated and proven below), we can write ϕ as a Boolean combination of formulas involving either only sort 1 or only sort 2. Since ϕ has only free variables of sort 1, every subformula involving sort 2 only can be evaluated independently of D: it is a sentence in the language of pure equality saying something about the complement of the active domain, which is isomorphic to U. We can therefore replace each subformula of ϕ involving only sort 2 by its

[2] R. Hull and J. Su. Domain independence and the relational calculus. *Acta Informatica* 31, 513–524 (1994).

absolute truth value on U, and obtain an equivalent formula φ mentioning only variables of sort 1. These variables range over the active domain only, so the theorem is proven.

Lemma 2 *Let σ be a vocabulary with the property that the sort symbols in σ can be partitioned into two parts, X and Y, such that the sort of every relation name of σ contains either only symbols from X, or from Y. Then every formula φ over σ is equivalent to a Boolean combination of formulas over σ which mention either only variables of sorts in X or of sorts in Y.*

Proof. By induction on φ.

- If φ is atomic and all variables occurring in it are either all of sorts in X or of sorts in Y, φ is already in the right form; otherwise, φ is equivalent to false by the assumption on σ.
- The cases φ is $(\neg\chi)$ or $(\chi \vee \theta)$ are trivial.
- If φ is $(\exists x)\chi$, we know by induction that χ can be written as a Boolean combination of the right form. Rewrite χ in disjunctive normal form and distribute the quantifier over the terms of the disjunction. We are left with a disjunction of terms of the form $(\exists x)(\psi_1 \wedge \cdots \wedge \psi_l \wedge \theta_1 \wedge \cdots \wedge \theta_m)$, where each ψ_i mentions only variables of sorts in X and each θ_i mentions only variables of sorts in Y. If x has sort in X, we rewrite the term as $(\exists x)(\psi_1 \wedge \cdots \wedge \psi_l) \wedge \theta_1 \wedge \cdots \wedge \theta_m$; if x has sort in Y, we rewrite as $\psi_1 \wedge \cdots \wedge \psi_l \wedge (\exists x)(\theta_1 \wedge \cdots \wedge \theta_m)$. Now φ is back in the right form.