On Two Forms of Structural Recursion

Dan Suciu¹ and Limsoon Wong²

¹ University of Pennsylvania, Philadelphia. Email: suciu@saul.cis.upenn.edu
² Institute of Systems Science, Singapore. Email: limsoon@iss.nus.sg

Abstract. We investigate and compare two forms of recursion on sets for querying nested collections. The first one is called sri and it corresponds to sequential processing of data. The second one is called sru and it corresponds to data-parallel processing. A uniform first-order translation from sru into sri was known from previous work. The converse translation is by necessity more difficult and we have obtained three main results concerning it. First, we exhibit a uniform translation of sri queries into sru queries over the nested relational algebra. We observe that this translation maps PTIME algorithms into exponential-space queries. The second result proves that any uniform translation of sri queries into sru queries over the nested relational algebra must map some PTIME queries into exponential-space ones. In fact, in the presence of certain external functions, we provide a PTIME sri query for which any equivalent sru query requires exponential space. Thus, as a mechanism for implementing algorithms over complex objects, sru is strictly less powerful than sri. This inefficiency is in contrast to a previous result that uniformly translates efficient sri programs into efficient sru programs, but over a language with higher-order functions. Our third result proves that, in the absence of external functions, higher-order functions do not add more expressive power to the nested relational algebra with sri or sru. However, elimination of higher-order functions cannot be done uniformly, because in the presence of certain external functions, more expressive power can be gained from the higher-order functions. These three results suggest that higher-order functions could be useful in query languages.

1 Introduction

Structural recursion is an attractive paradigm for programming with sets. It can be conceived in two distinct ways, corresponding to the two distinct universal properties that the finite set construction enjoys [7]. The first gives rise to a form of structural recursion that Tannen, Buneman, and Wong call sri [8]. It is closely related to the set_reduce of Stemple and Sheard [19] and the fold operator in functional programming languages [5]. Essentially, sri defines a function $g$ on a set $O$ by iteration in an order-independent manner over the elements of $O$. Namely, it allows us to define $g$ by:

$$
g(\{\}) = e
$$
$$
g(\{x\} \cup O) = i(x, g(O))
$$
Here $i$ is a previously defined function and $e$ is some value. We write $sri(i, e)$ to denote the function $g$ thus defined. In order for $g$ to be well defined, $i$ has to satisfy the following two conditions: $i(x, i(y, a)) = i(y, i(x, a))$ (commutativity), and $i(x, i(x, a)) = i(x, a)$ (idempotence). This form of recursion is related to the primitive recursion schema in the theory of recursive functions [16].

The second form of structural recursion is called $sru$ by Tannen, Buneman, and Wong [8], from structural recursion on the union presentation. It is related to the hom operator of Machiavelli [17] and the pump operator of FAD [4]. Essentially, $sru$ allows us to define a function $h$ on a set $O$ in a divide-and-conquer manner. To compute $h$, one has to divide $O$ recursively in an order-independent way into ever smaller subsets until one reaches sets with 0 or 1 elements. More precisely, $sru$ allows us to define some function $h$ by:

$$
\begin{align*}
    h(\{\}) &= e \\
    h(\{x\}) &= f(x) \\
    h(O_1 \cup O_2) &= u(h(O_1), h(O_2))
\end{align*}
$$

We use $sru(u, f, e)$ to denote the $h$ thus defined. Note that it is not well defined unless the following conditions are satisfied: $u(e, a) = u(a, e) = a$ (identity), $u(a, b) = u(b, a)$ (commutativity), and $u(a, a) = a$ (idempotence).

When $sri$ and $sru$ are considered as candidates to be incorporated in a database query language, it is necessary to compare the relative expressive power of the two resultant languages. A straightforward and efficient translation of $sru$ into $sri$ is described in Tannen and Subrahmanym [7]. This translation is first-order expressible. This implies that any reasonable language augmented with $sri$ is at least as powerful as the same language augmented with $sru$. Furthermore, any efficient algorithm expressed in the language with $sru$ is translated into an efficient one in the language with $sri$. Tannen and Subrahmanym also give a simple translation of $sri$ into $sru$. However, this translation uses higher-order functions. This paper investigates translations of $sri$ into $sru$ over query languages for complex objects without higher-order functions.

We use the nested relational calculus $\mathcal{N}RC$ of Wong [23] as the ambient query language in this investigation. $\mathcal{N}RC$ has the same power as the query languages for nested relations of Thomas and Fischer [22], Schek and Scholl [18], Colby [9], etc. We add the two forms of structural recursion above to $\mathcal{N}RC$. Our first result is a uniform translation of queries in $\mathcal{N}RC(sri)$ into queries in $\mathcal{N}RC(sru)$. This translation works in the presence of arbitrary external functions. However, the translation is expensive.

Our second result proves that any uniform translation of queries from $\mathcal{N}RC(sri)$ into $\mathcal{N}RC(sru)$ has to be expensive. In particular, any such uniform translation must map some PTIME algorithms into exponential space ones. It should be stressed that this result does not compare the relative expressive power of the two languages, which is the same. Rather, this result compares their ability to express efficient algorithms. It is in the same spirit as that of Abiteboul and Vianu [3], proving that parity cannot be expressed in PTIME by a Generic Machine, and as that of Suciu and Paredaens [21], proving that Abiteboul and