A decision procedure for term algebras with queues

Tatiana Rybina[*] Andrei Voronkov[‡]

July 14, 2000

Abstract

In software verification it is often required to prove statements about heterogeneous domains containing elements of various sorts, such as counters, stacks, lists, trees and queues. Any domain with counters, stacks, lists, and trees (but not queues) can be easily seen a special case of the term algebra, and hence a decision procedure for term algebras can be applied to decide the first-order theory of such a domain.

We prove that the first-order theory of term algebras extended with queues is decidable by presenting a quantifier-elimination procedure for this theory.

Term algebras with queues

Sorts. We assume a finite set of *basic sort*. We define *sort* as follows: (i) every basic sort is a sort; (ii) if α is a sort, then $queue(\alpha)$ is a sort, called a *queue sort*.

We call a *function type* any expression of the form $\alpha_1 \times \ldots \times \alpha_n \to \beta$, where $\alpha_1, \ldots, \alpha_n, \beta$ are sorts, $n \geq 0$. When $n = 0$, we will write β instead of $\to \beta$. A *signature* $\mathcal{S}$ is a pair consisting of a finite set of function symbols, and a function : mapping every function symbol f to a function type $\alpha_1 \times \ldots \times \alpha_n \to \beta$, called the *type* of f, where β is a basic sort. The number n is called the *arity* of f. Function symbols of arity 0 are called *constants*.

Structure $\mathcal{Q}(\mathcal{S})$. Every signature $\mathcal{S}$ defines a unique structure $\mathcal{Q}(\mathcal{S})$, called a *term algebra with queues* as follows.

The *universe* of $\mathcal{Q}(\mathcal{S})$ is defined inductively as follows.

1. For every constant $a : \beta$, a is an element of $\mathcal{Q}(\mathcal{S})$ of the sort β.

2. For every elements $e_1, \ldots, e_n$ of the same sort α, where $n \geq 0$, the sequence $e_1 \ldots e_n$ is an element of $\mathcal{Q}(\mathcal{S})$ of the sort $queue(\alpha)$. Elements of any sort $queue(\alpha)$ will be called *queues*. The empty sequence is called the *empty queue*, and denoted ε_α.

3. If $f : \alpha_1 \times \ldots \times \alpha_n \to \beta$ and $e_1, \ldots, e_n$ are elements of $\mathcal{Q}(\mathcal{S})$ of the sorts $\alpha_1, \ldots, \alpha_n$, respectively, then $f(e_1, \ldots, e_n)$ is an element of $\mathcal{Q}(\mathcal{S})$ of the sort β.

We consider two elements equal only if they have the same sort and coincide as expressions. Using different signatures we obtain different domains, in which sorts can be finite or infinite. For simplicity, we assume that each basic sort has a nonempty domain.

[*]Partially supported by a grant from EPSRC (Rainbow project).

[†]Partially supported by grants from EPSRC and the Faculty of Science and Engineering.

[‡]This abstract is based on [Rybina and Voronkov 2000].

The *language of* $\mathcal{Q}(\mathcal{S})$, denoted $\mathcal{L}_{\mathcal{Q}(\mathcal{S})}$, is a first-order language using, in addition to function symbols in $\mathcal{S}$, the equality relation symbol $=$ and the following symbols: (i) for every sort α, the constant $\varepsilon_\alpha : queue(\alpha)$; (ii) for every sort α, the function symbols $ladd_\alpha : \alpha \times queue(\alpha) \to queue(\alpha)$ and $radd_\alpha : \alpha \times queue(\alpha) \to queue(\alpha)$.

Semantics of $\mathcal{Q}(\mathcal{S})$. For every function symbol $f : \alpha_1 \times \ldots \times \alpha_n \to \beta$ of $\mathcal{S}$, the interpretation $\underline{f}$ of this symbol in $\mathcal{Q}(\mathcal{S})$ is defined as follows: for all elements $e_1, \ldots, e_n$ of the sorts $\alpha_1, \ldots, \alpha_n$ we have $\underline{f}(e_1, \ldots, e_n) = f(e_1, \ldots, e_n)$. Likewise, the interpretation $\underline{\varepsilon_\alpha}$ of the constant ε_α is the empty queue of the sort $queue(\alpha)$. The function symbols $\underline{ladd_\alpha}$ and $\underline{radd_\alpha}$ of the function symbols $ladd_\alpha$ and $radd_\alpha$ are interpreted as the left- and right- addition of an element to a queue.

Quantifier elimination

Similar to Kunen [1987] and Belegradek [1988], we extend the language $\mathcal{L}_{\mathcal{Q}(\mathcal{S})}$ to a new language, denoted $\mathcal{L}'_{\mathcal{Q}(\mathcal{S})}$, by introducing the following collection of decomposition function symbols and relation symbols.

1. A unary relation symbol $\mathbf{Is}_f : \beta$ for each function symbol $f : \alpha_1 \times \ldots \times \alpha_n \to \beta$, $n \geq 1$ of $\mathcal{S}$. This symbols is interpreted as follows: $\mathbf{Is}_f(a)$ is true if and only if a has the form $f(\ldots)$.

2. A unary function symbol $f_i : \beta \to \alpha_i$, for each function symbol $f : \alpha_1 \times \ldots \times \alpha_n \to \beta$, $n \geq 1$ of $\mathcal{S}$ and each $i = 1, \ldots, n$. We have $f_i(f(t_1, \ldots, t_n)) = t_i$.

3. unary function symbols

$$\begin{array}{ll} lhead_\alpha : queue(\alpha) \to \alpha & rhead_\alpha : queue(\alpha) \to \alpha \\ ltail_\alpha : queue(\alpha) \to queue(\alpha) & rtail_\alpha : queue(\alpha) \to queue(\alpha) \end{array}$$

   for each sort α. For every queue q, $lhead(q)$ gives the leftmost element of q, and $ltail(q)$ is the queue obtained by removing this element, and similar for *rhead* and *rtail*.

THEOREM 1 (Quantifier Elimination) *The first-order theory of term algebras with queues in the language $\mathcal{L}'_{\mathcal{Q}(\mathcal{S})}$ admits quantifier elimination.*

COROLLARY 2 (Decidability) *The first-order theory of any term algebra with queues is decidable.*

COROLLARY 3 (Words) *The first-order theory of words with the operations of the left and right multiplications by a letter is decidable.*

The monadic second-order theory of this structure is known to be undecidable.

References

BELEGRADEK O. [1988], Model theory of locally free algebras (in Russian), *in* 'Model Theory and its Applications', Vol. 8 of *Trudy Instituta Matematiki*, Nauka, Novosibirsk, pp. 3–24.

KUNEN K. [1987], 'Negation in logic programming', *Journal of Logic Programming* **4**, 289–308.

RYBINA T. AND VORONKOV A. [2000], A decision procedure for term algebras with queues, *in* 'Proc. 15th Annual IEEE Symp. on Logic in Computer Science', Santa Barbara, California, pp. 279–290.