An Upper Bound for the Equational Specification of Finite State Services

J.A. Bergstra
I. Bethke
An upper bound for the equational specification of finite state services

J.A. Bergstra\textsuperscript{a, b, 1} I. Bethke\textsuperscript{b, 2}

\textsuperscript{a}Utrecht University, Department of Philosophy, Applied Logic Group
\textsuperscript{b}University of Amsterdam, Faculty of Science, Programming Research Group

\textit{Key words:} client-server composition, equational specification, finite data types, boundedness properties.

1 Introduction

The communication mechanism between client and server tasks can be modelled by a simple rendezvous model. The intuition here is that a process—the client—when executed places its requests into a request buffer. These requests are taken from the buffer by another process—the server. After some processing, prescribed by actions of the server, the server returns a Boolean reply indicating success or failure of the request.

In this note in the honour of John-Jules Meyer’s 50th birthday we shall describe the algebra of \textit{state services} which corresponds to the server side of the communication model. We shall prove a theorem concerning the boundedness of the number of equations needed to specify \textit{finite minimal state services} within the mathematical framework of the initial algebra semantics for data types. This theorem is inspired by earlier results on boundedness properties of equational specifications by the first author and John-Jules Meyer in [3] and [4]. For general considerations on this subject we refer to [2].

\begin{itemize}
  \item \textsuperscript{1} E-mail: Jan.Bergstra@phil.uu.nl
  \item \textsuperscript{2} E-mail: inge@science.uva.nl
\end{itemize}
State service algebras are algebras with three domains: $\mathbb{S}$ (services), $\mathbb{S}_t$ (states) and $\mathbb{B}$ (Booleans). There are five constants $\delta, \epsilon \in \mathbb{S}$ (deadlock and the empty service), $T, F \in \mathbb{B}$ and the initial state $S_0 \in \mathbb{S}_t$, a unary service function $S : \mathbb{S}_t \to \mathbb{S}$, two binary functions $\leftrightarrow, \cdot : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ (preferential choice and sequential composition), and a ternary function $\triangleright \triangleleft : \mathbb{S} \times \mathbb{B} \times \mathbb{S}$ (if $b$ then $S$ else $S'$). The rest of the signature is generated by a set $A$ of actions: there are constants $a+, a- \in \mathbb{S}$ denoting the service consisting of the processing of action $a$ followed by a positive or negative reply, a unary mapping $\text{effect}_a : \mathbb{S}_t \to \mathbb{S}_t$ turning a state into the state obtained after the execution of $a$, and a unary function $\text{reply}_a : \mathbb{S}_t \to \mathbb{B}$ assigning to each state the Boolean reply returned after the action $a$ is carried out. More formally,

**Definition 2.1** Let $A$ be a set of actions.

1. The signature of state service algebras is $\Sigma_A = (S, \Omega_A)$ with the set of sorts $S = \{\mathbb{S}, \mathbb{S}_t, \mathbb{B}\}$ and the set of operations

$$\Omega_A = \{\delta, \epsilon : \mathbb{S}, T, F : \mathbb{B}, S_0 : \mathbb{S}_t\}$$

$$\cup \{S : \mathbb{S}_t \to \mathbb{S}, \leftrightarrow, \cdot : \mathbb{S} \times \mathbb{S} \to \mathbb{S}, \triangleright \triangleleft : \mathbb{S} \times \mathbb{B} \times \mathbb{S}\}$$

$$\cup \{a+, a- : \mathbb{S}, \text{effect}_a : \mathbb{S}_t \to \mathbb{S}_t, \text{reply}_a : \mathbb{S}_t \to \mathbb{B} \mid a \in A\}$$

Notice that the reduct $\langle \{\mathbb{S}_t, \mathbb{B}\}, \{S_0 : \mathbb{S}_t\} \cup \{\text{effect}_a : \mathbb{S}_t \to \mathbb{S}_t, \text{reply}_a : \mathbb{S}_t \to \mathbb{B} \mid a \in A\}$ is a particular two-sorted monoid which was coined process algebra in [3]. In later work of the first author, e.g. [1], this term is used in a different setting.

2. A state service algebra $\mathbb{S}$ is a $\Sigma_A$ algebra satisfying the axioms

$$\epsilon \cdot X = X$$
$$X \cdot \epsilon = X$$
$$\delta \cdot X = \delta$$
$$\delta \leftrightarrow X = X$$
$$X \leftrightarrow \delta = \delta$$
$$(X \leftrightarrow Y) \leftrightarrow Z = X \leftrightarrow (Y \leftrightarrow Z)$$
$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$
$$(X \leftrightarrow Y) \cdot Z = (X \cdot Z) \leftrightarrow (Y \cdot Z)$$
$$X \triangleleft T \triangleright Z = X$$
$$X \triangleleft F \triangleright Z = Z$$

2
a+ · X ↔ a+ · Y = a+ · X
a+ · X ↔ a− · Y = a+ · X
a− · X ↔ a+ · Y = a− · X
a− · X ↔ a− · Y = a− · X
a ≠ b ⇒ a+ · X ↔ b+ · Y = b+ · Y ↔ a+ · X
a ≠ b ⇒ a+ · X ↔ b− · Y = b− · Y ↔ a+ · X
a ≠ b ⇒ a− · X ↔ b+ · Y = b+ · Y ↔ a− · X
a ≠ b ⇒ a− · X ↔ b− · Y = b− · Y ↔ a− · X

Note that the preferential choice is deterministic on services performing the same action. We say that $S$ is finite if $S$ is finite, and minimal if $S$ is generated by the initial state $S_0$ and the functions $\text{effect}_a$ with $a \in A$.

We give three examples of state service algebras.

**Examples 2.2**

(1) The first example is the finite state service of a single boolean value that can be set and read. We let $S\text{.bool}$ be the state service algebra with the two states 0 and 1 and $A = \{ set : 0, set : 1, eq : 0, eq : 1 \}$ and

$$S(0) = set : 0+ \cdot S(0) \leftrightarrow set : 1+ \cdot S(1) \leftrightarrow eq : 0+ \cdot S(0) \leftrightarrow eq : 1- \cdot S(0)$$

$$S(1) = set : 0+ \cdot S(0) \leftrightarrow set : 1+ \cdot S(1) \leftrightarrow eq : 0- \cdot S(1) \leftrightarrow eq : 1+ \cdot S(1)$$

(2) As a second example we consider the infinite natural number counter with a test on 0 and increment and decrement actions. Here we have the states 0, 1, 2, ..., $A = \{eq : 0, inc, dec\}$ and

$$S(0) = eq : 0+ \cdot S(0) \leftrightarrow inc+ \cdot S(1) \leftrightarrow dec- \cdot S(0)$$

$$S(n + 1) = eq : 0- \cdot S(n + 1) \leftrightarrow inc+ \cdot S(n + 2) \leftrightarrow dec+ \cdot S(n)$$

(3) Our last example specifies a stack of Booleans with a test on emptiness and the top value, and push and pop actions. Here $S = \{ \sigma \mid \sigma \in \{0, 1\}^* \}$, $A = \{ isempty, push : 0, push : 1, pop, topeq : 0, topeq : 1 \}$ and

$$S(<>) = isempty+ \cdot S(<>) \leftrightarrow push : 0+ \cdot S(0) \leftrightarrow push : 1+ \cdot S(1)$$

$$\leftrightarrow pop- \cdot S(<>) \leftrightarrow topeq : 0- \cdot S(<>) \leftrightarrow topeq : 1- \cdot S(<>)$$

$$S(\sigma 0) = isempty- \cdot S(\sigma 0) \leftrightarrow push : 0+ \cdot S(\sigma 0) \leftrightarrow push : 1+ \cdot S(\sigma 0)$$

$$\leftrightarrow pop+ \cdot S(\sigma) \leftrightarrow topeq : 0+ \cdot S(\sigma 0) \leftrightarrow topeq : 1- \cdot S(\sigma 0)$$

$$S(\sigma 1) = isempty- \cdot S(\sigma 1) \leftrightarrow push : 0+ \cdot S(\sigma 1) \leftrightarrow push : 1+ \cdot S(\sigma 1)$$

$$\leftrightarrow pop+ \cdot S(\sigma) \leftrightarrow topeq : 0- \cdot S(\sigma 1) \leftrightarrow topeq : 1+ \cdot S(\sigma 1)$$
3 The theorem

Our theorem gives an upper bound for the number of equations needed to specify the service function for finite minimal service algebras in terms of the number of its actions.

**Theorem 3.1** Let $A = \{a_1, \ldots, a_k\}$ be a set of $k$ actions and $S$ be a finite minimal state service algebra. Then $S$ can be specified involving at most $14 + 2k$ auxiliary unary functions and $92 + 5k$ equations.

**Proof.** Let $A' = \{a \mid a \in A \& \text{reply}^{-1}_a(T) \neq \emptyset \neq \text{reply}^{-1}_a(F)\}$. Choose for every $a \in A'$ functions $t_a : S \rightarrow \text{reply}^{-1}_a(T)$ and $f_a : S \rightarrow \text{reply}^{-1}_a(F)$ such that

- $\forall x \in \text{reply}_a^{-1}(T) \ t_a(x) = x$, and
- $\forall x \in \text{reply}_a^{-1}(F) \ f_a(x) = x$.

Note that we then have

- $\forall x \ \text{reply}_a(x) = T \Leftrightarrow t_a(x) = x$, and
- $\forall x \ \text{reply}_a(x) = F \Leftrightarrow f_a(x) = x$.

By the theorem in [3] we can specify the unoid

$$(S, \{S_0\} \cup \{\text{effect}_a \mid a \in A\} \cup \{t_a, f_a \mid a \in A'\})$$

by 12 hidden individual constants, 13 hidden unary functions and $90 + (1 + k + 2 \times |A'|)$ equations. We replace the individual constants by an unary enumeration function. We can now specify the reply function by the equations

\[
\begin{align*}
\text{reply}_a(x) &= T \text{ if } \text{reply}_a(S) = \{T\} \\
\text{reply}_a(x) &= F \text{ if } \text{reply}_a(S) = \{F\} \\
\text{reply}_a(t_a(x)) &= T \text{ if } a \in A' \\
\text{reply}_a(f_a(x)) &= F \text{ if } a \in A'
\end{align*}
\]

This amounts to another $|A - A'| + 2 \times |A'|$ equations. The last equation is

\[
S(x) = a_1 + \cdot S(\text{effect}_{a_1}(x)) < \text{reply}_{a_1}(x) \triangleright a_1 - \cdot S(\text{effect}_{a_1}) \\
\leftrightarrow \quad \cdots \\
\vdots \\
\leftrightarrow \quad \cdots \\
\leftrightarrow a_k + \cdot S(\text{effect}_{a_k}(x)) < \text{reply}_{a_k}(x) \triangleright a_k - \cdot S(\text{effect}_{a_k}(x))
\]
Adding up we arrive at
\[ 90 + 1 + k + |A - A'| + 4 \times |A'| + 1 \leq 92 + 5k \]
equations. □

**Remark 3.2** In a process algebra setting such as in [1] the specification of
the service function \( S \) can be given in a slightly more compact way by

\[
S(x) = r(a_1) \cdot s(\text{reply}_{a_1}) \cdot S(\text{effect}_{a_1}(x)) \\
+ \cdots \\
\vdots \\
+ \cdots \\
+ r(a_k) \cdot s(\text{reply}_{a_k}) \cdot S(\text{effect}_{a_k}(x))
\]

with read and send actions \( r(a_1), \ldots, r(a_k), s(\text{reply}_{a_1}), \ldots, s(\text{reply}_{a_k}) \). Here,
however, choice will be non-deterministic.

**References**


Electronic Reports Series of the Programming Research Group

Within this series the following reports appeared.


The above reports and more are available through the website: www.science.uva.nl/research/prog/