University of Amsterdam
Programming Research Group

An Upper Bound for the Equational Specification of Finite State Services

J.A. Bergstra
I. Bethke

Report PRG0405 September 2004

J.A. Bergstra

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
The Netherlands

tel. +31 20 525.7591
e-mail: janb@science.uva.nl

I. Bethke

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
The Netherlands

tel. +31 20 525.7583
e-mail: inge@science.uva.nl

Programming Research Group Electronic Report Series

An upper bound for the equational specification of finite state services

J.A. Bergstra [a,b,1] I. Bethke [b,2]

[a] *Utrecht University, Department of Philosophy, Applied Logic Group*
[b] *University of Amsterdam, Faculty of Science, Programming Research Group*

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 *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 *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].

[1] E-mail: Jan.Bergstra@phil.uu.nl
[2] E-mail: inge@science.uva.nl

2 Equational specification of services

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 $\leftarrow\!\!+\;, \cdot : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ (preferential choice and sequential composition), and a ternary function $\_ \lhd \_ \rhd \_ : \mathbb{S} \times \mathbb{B} \times \mathbb{S}$ (*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 $\mathit{effect}_a : \mathbb{S}t \to \mathbb{S}t$ turning a state into the state obtained after the execution of a, and a unary function $\mathit{reply}_a : \mathbb{S}t \to \mathbb{B}$ 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

$$\begin{aligned}\Omega_A = &\{\delta, \epsilon : \mathbb{S}, T, F : \mathbb{B}, S_0 : \mathbb{S}t\}\\ &\cup\{S : \mathbb{S}t \to \mathbb{S}, \leftarrow\!\!+\;, \cdot : \mathbb{S} \times \mathbb{S} \to \mathbb{S}, \_ \lhd \_ \rhd \_ : \mathbb{S} \times \mathbb{B} \times \mathbb{S}\}\\ &\cup\{a+, a- : \mathbb{S}, \mathit{effect}_a : \mathbb{S}t \to \mathbb{S}t, \mathit{reply}_a : \mathbb{S}t \to \mathbb{B} \mid a \in A\}\end{aligned}$$

Notice that the reduct $(\{\mathbb{S}t, \mathbb{B}\}, \{S_0 : \mathbb{S}t\} \cup \{\mathit{effect}_a : \mathbb{S}t \to \mathbb{S}t, \mathit{reply}_a : \mathbb{S}t \to \mathbb{B} \mid a \in A\}$ 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 Σ_A algebra satisfying the axioms

$$\begin{aligned}\epsilon \cdot X &= X\\ X \cdot \epsilon &= X\\ \delta \cdot X &= \delta\\ \delta \leftarrow\!\!+\; X &= X\\ X \leftarrow\!\!+\; \delta &= \delta\\ (X \leftarrow\!\!+\; Y) \leftarrow\!\!+\; Z &= X \leftarrow\!\!+\; (Y \leftarrow\!\!+\; Z)\\ (X \cdot Y) \cdot Z &= X \cdot (Y \cdot Z)\\ (X \leftarrow\!\!+\; Y) \cdot Z &= (X \cdot Z) \leftarrow\!\!+\; (Y \cdot Z)\\ X \lhd T \rhd Z &= X\\ X \lhd F \rhd Z &= Z\end{aligned}$$

$$a+ \cdot X \leftarrow\!\!+\ a+ \cdot Y = a+ \cdot X$$
$$a+ \cdot X \leftarrow\!\!+\ a- \cdot Y = a+ \cdot X$$
$$a- \cdot X \leftarrow\!\!+\ a+ \cdot Y = a- \cdot X$$
$$a- \cdot X \leftarrow\!\!+\ a- \cdot Y = a- \cdot X$$
$$a \neq b \Rightarrow a+ \cdot X \leftarrow\!\!+\ b+ \cdot Y = b+ \cdot Y \leftarrow\!\!+\ a+ \cdot X$$
$$a \neq b \Rightarrow a+ \cdot X \leftarrow\!\!+\ b- \cdot Y = b- \cdot Y \leftarrow\!\!+\ a+ \cdot X$$
$$a \neq b \Rightarrow a- \cdot X \leftarrow\!\!+\ b+ \cdot Y = b+ \cdot Y \leftarrow\!\!+\ a- \cdot X$$
$$a \neq b \Rightarrow a- \cdot X \leftarrow\!\!+\ b- \cdot Y = b- \cdot Y \leftarrow\!\!+\ a- \cdot X$$

Note that the preferential choice is deterministic on services performing the same action. We say that $\mathbb{S}$ is *finite* if $\mathbb{S}t$ is finite, and *minimal* if $\mathbb{S}t$ is generated by the initial state S_0 and the functions $effect_a$ 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 $\mathbb{S}\_bool$ be the state service algebra with the two states 0 and 1 and $A = \{set:0, set:1, eq:0, eq:1\}$ and

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

(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 $0, 1, 2, \ldots$, $A = \{eq:0, inc, dec\}$ and

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

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

$$\begin{aligned}
S(<>) &= isempty+ \cdot S(<>) \leftarrow\!\!+\ push:0+ \cdot S(0) \leftarrow\!\!+\ push:1+ \cdot S(1) \\
&\quad \leftarrow\!\!+\ pop- \cdot S(<>) \leftarrow\!\!+\ topeq:0- \cdot S(<>) \leftarrow\!\!+\ topeq:1- \cdot S(<>) \\
S(\sigma 0) &= isempty- \cdot S(\sigma 0) \leftarrow\!\!+\ push:0+ \cdot S(\sigma 00) \leftarrow\!\!+\ push:1+ \cdot S(\sigma 01) \\
&\quad \leftarrow\!\!+\ pop+ \cdot S(\sigma) \leftarrow\!\!+\ topeq:0+ \cdot S(\sigma 0) \leftarrow\!\!+\ topeq:1- \cdot S(\sigma 0) \\
S(\sigma 1) &= isempty- \cdot S(\sigma 1) \leftarrow\!\!+\ push:0+ \cdot S(\sigma 10) \leftarrow\!\!+\ push:1+ \cdot S(\sigma 11) \\
&\quad \leftarrow\!\!+\ pop+ \cdot S(\sigma) \leftarrow\!\!+\ topeq:0- \cdot S(\sigma 1) \leftarrow\!\!+\ topeq:1+ \cdot S(\sigma 1)
\end{aligned}$$

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 $\mathbb{S}$ 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 $A' = \{a \mid a \in A \ \& \ reply_a^{-1}(T) \neq \emptyset \neq reply_a^{-1}(F)\}$. Choose for every $a \in A'$ functions $t_a : \mathbb{S}t \to reply_a^{-1}(T)$ and $f_a : \mathbb{S}t \to reply_a^{-1}(F)$ such that

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

Note that we then have

- $\forall x\ reply_a(x) = T \Leftrightarrow t_a(x) = x$, and
- $\forall x\ reply_a(x) = F \Leftrightarrow f_a(x) = x$.

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

$$(\mathbb{S}t, \{S_0\} \cup \{effect_a \mid a \in A\} \cup \{t_a, f_a \mid a \in A'\})$$

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

$$\begin{aligned}
reply_a(x) &= T \text{ if } reply_a(\mathbb{S}t) = \{T\} \\
reply_a(x) &= F \text{ if } reply_a(\mathbb{S}t) = \{F\} \\
reply_a(t_a(x)) &= T \text{ if } a \in A' \\
reply_a(f_a(x)) &= F \text{ if } a \in A'
\end{aligned}$$

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

$$\begin{aligned}
S(x) = {} & a_1{+} \cdot S(effect_{a_1}(x)) \lhd reply_{a_1}(x) \rhd a_1{-} \cdot S(effect_{a_1}) \\
& \leftarrow\!\!+ \ \cdots \\
& \vdots \\
& \leftarrow\!\!+ \ \cdots \\
& \leftarrow\!\!+ \ a_k{+} \cdot S(effect_{a_k}(x)) \lhd reply_{a_k}(x) \rhd a_k{-} \cdot S(effect_{a_k}(x))
\end{aligned}$$

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

$$\begin{aligned} S(x) = {} & r(a_1) \cdot s(reply_{a_1}) \cdot S(effect_{a_1}(x)) \\ & + \cdots \\ & \vdots \\ & + \cdots \\ & + r(a_k) \cdot s(reply_{a_k}) \cdot S(effect_{a_k}(x)) \end{aligned}$$

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

References

[1] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.

[2] J.A. Bergstra, S. Mauw and F. Wiedijk. Uniform Algebraic Specifications of Finite Sets with Equality. *International Journal of Foundations of Computer Science*, 2(1):43–65, 1991.

[3] J.A. Bergstra and J.-J.CH. Meyer. *On Axiomatising Finite Data Structures*. Rapport No. 80–18, Institute of Applied Mathematics and Computer Science, University of Leiden, Leiden, The Netherlands, 1980.

[4] J.A. Bergstra and J.-J.CH. Meyer. The Equational Specification of Finite Minimal Unoids Using Only Unary Hidden Functions. *Fundamenta Informaticae*, 2:143–170, 1982.

Within this series the following reports appeared.

[PRG0405] J.A. Bergstra and I. Bethke, *An Upper Bound for the Equational Specification of Finite State Services*, Programming Research Group - University of Amsterdam, 2004.

[PRG0404] J.A. Bergstra and C.A. Middelburg, *Thread Algebra for Strategic Interleaving*, Programming Research Group - University of Amsterdam, 2004.

[PRG0403] B. Diertens, *A Compiler-projection from PGLEc.MSPio to Parrot*, Programming Research Group - University of Amsterdam, 2004.

[PRG0402] J.A. Bergstra and I. Bethke, *Linear Projective Program Syntax*, Programming Research Group - University of Amsterdam, 2004.

[PRG0401] B. Diertens, *Molecular Scripting Primitives*, Programming Research Group - University of Amsterdam, 2004.

[PRG0302] B. Diertens, *A Toolset for PGA*, Programming Research Group - University of Amsterdam, 2003.

[PRG0301] J.A. Bergstra and P. Walters, *Projection Semantics for Multi-File Programs*, Programming Research Group - University of Amsterdam, 2003.

[PRG0201] I. Bethke and P. Walters, *Molecule-oriented Java Programs for Cyclic Sequences*, Programming Research Group - University of Amsterdam, 2002.

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

Electronic Report Series

Programming Research Group
Faculty of Science
University of Amsterdam

Kruislaan 403
1098 SJ Amsterdam
the Netherlands

www.science.uva.nl/research/prog/