Substitution in Structural Operational Semantics and value-passing process calculi

Sam Staton

Computer Laboratory
University of Cambridge

Abstract

Consider a process calculus that allows agents to communicate values. The structural operational semantics involves substitution of values for variables. Existing rule formats, such as the GSOS format, do not allow this kind of explicit substitution in the semantic rules. We investigate how to derive rule formats for languages with substitution, by using categorical logic to interpret the framework of the GSOS format in different categories. The categories in question are categories of 'substitution actions'.

1 A simple language for value-passing

To set the scene, fix a set of channel names, and consider a set $\mathbb{V}$ of value-expressions, that includes the channel names. A simple untyped value-passing process language, $\mathbb{V}$-CCS, is given in Figure 1 (c.f. [8]).

The precise value expressions of $\mathbb{V}$ are not important, but note that since $\mathbb{V}$ includes the (static) channel names, $\mathbb{V}$-CCS is a very primitive applied π-calculus without restriction or name generation; c.f. [1]. For the sake of illustration, consider the set $\mathbb{V}_{\mathrm{ex}}$ of value expressions determined by the following grammar:

$$v \ ::= \ n \mid v+v \mid (v,v) \mid \pi_1(v) \mid \pi_2(v) \mid c \qquad (n \text{ is a number, } c \text{ is a channel name}).$$

We will always work with value expressions up-to the evident equations ($2+3=5$; $\pi_1(v,w)=v$; etc.), rather than explicitly evaluating or normalizing them; this is to simplify the presentation. The following transitions are derivable in $\mathbb{V}_{\mathrm{ex}}$-CCS.

$$(\bar{c}\langle 3\rangle.\mathbf{0}) \,|\, (c(v).\bar{c}\langle 2+v\rangle.\mathbf{0}) \ \xrightarrow{\tau}\ \mathbf{0} \,|\, \bar{c}\langle 2+3\rangle.\mathbf{0} \ \xrightarrow{\bar{c}\langle 5\rangle}\ \mathbf{0}\,|\,\mathbf{0}$$

$$P \ ::= \ \mathbf{0} \mid P\,|\,P \mid v(a).P \mid \bar{v}\langle w\rangle.P \quad (v,w \in \mathbb{V})$$

$$\frac{}{c(a).P \xrightarrow{c\langle v\rangle} \{^{v}/_{a}\}P}\,(\textit{input}) \qquad \frac{}{\bar{c}\langle v\rangle.P \xrightarrow{\bar{c}\langle v\rangle} P}\,(\textit{output})$$

$$\frac{P \xrightarrow{l} P'}{P\,|\,Q \xrightarrow{l} P'\,|\,Q}\,(\textit{parallel}) \qquad \frac{P \xrightarrow{c\langle v\rangle} P' \quad Q \xrightarrow{\bar{c}\langle v\rangle} Q'}{P\,|\,Q \xrightarrow{\tau} P'\,|\,Q'}\,(\textit{communication})$$

Fig. 1. A simple value-passing language, $\mathbb{V}$-CCS. We elide symmetric versions of the rules (*parallel*) and (*communication*). The set $\mathbb{V}$ of values is assumed to contain a set of channel names, and the (*input*), (*output*) and (*communication*) rules carry a side condition that c is a channel name.

2 Value-passing systems and the GSOS rule format

The GSOS rule format was introduced by [2]. A transition system specification is in the *positive GSOS format* if it is specified by rules of the following form:

$$\frac{(x_{i_j} \xrightarrow{l} y_j \mid 1 \leq j \leq m)}{o(x_1, \dots, x_n) \xrightarrow{L} t}$$

where the x_i's and y_j's are all different, and the only variables appearing in t are the x_i's and y_j's.

There are various things one can say about a language semantics if it is specified in the GSOS format; most interesting is that bisimilarity ($\sim$) is a congruence:

$$\text{if } x_1 \sim x'_1, \dots, \text{ and } x_n \sim x'_n, \text{ then } o(x_1, \dots, x_n) \sim o(x'_1, \dots, x'_n).$$

Value-passing calculi are not GSOS (well, not classically). The language $\mathbb{V}$-CCS is roughly in the shape of the GSOS format, but does not fit properly into the framework. The main problems arise in the (*input*) rule, which includes (i) a variable a that is binding in P, and (ii) a substitution $\{^v/_a\}P$. Neither of these features are permitted in the GSOS format.

A third problem is that it is natural to consider $\mathbb{V}$-CCS terms with free value-variables, in order to define a notion of congruence that respects input contexts. For this reason we recall a more elaborate notion of bisimulation; c.f. [9]:

Definition 1. A bisimulation relation R on open $\mathbb{V}$-CCS terms is an *open bisimulation* if it is closed under substitution: if $P\,R\,Q$ then $(\{^v/_a\}P)\ R\ (\{^v/_a\}Q)$. *Open bisimilarity* is the greatest open bisimulation.

Value-passing calculi *are* GSOS, categorically. In the remainder of this note, I sketch how the specifications of value-passing calculi can be seen as GSOS specifications, by working in a category of *substitution actions*, rather than the category of sets and functions. We use the techniques introduced in [10] for name-passing calculi.

3 GSOS in type theory

To make things slightly more general and abstract, we reformulate the structure and requirements of the positive GSOS format in more fundamental terms. We only summarize the developments here; more details are in [10], where the more general tyft/tyxt format is considered.

Syntax and signatures. Traditionally, a first order signature is a set of operators, together with an arity for each operator. It is helpful to allow the arities of operators to be arbitrary sets, and not just natural numbers. In the notation of type theory: we have a set O of operators and a function $O \to \mathtt{Set}$ assigning an arity to each operator. This is a generalization, since individual natural numbers can be thought of as sets $[n] = \{1 \dots n\}$.

The usual concepts of algebra and congruence can be defined for this notion of signature, and the free algebra of terms can be built. For a signature (O, A)

```
record Transition-system-specification where
    field O          : Set          -- a signature: a set of operators
          A          : O → Set      -- . . . and arities for each operator
          Label      : Set          -- a set of labels
          Rule       : Set          -- a set of rules
          Prem       : Rule → Set   -- a set of premises for each rule
          Var        : Rule → Set   -- a set of variables for each rule

            -- structure of the premises for each rule
          Prem-lhs  : (r : Rule) → Prem r → Var r          -- a variable on either side
          Prem-rhs  : (r : Rule) → Prem r → Var r
          Prem-lab  : (r : Rule) → Prem r → Label          -- a label on each premise

            -- structure of the conclusion, for each rule
          Con-lhs-o : Rule → O                             -- an operator on the l.h.s.
          Con-lhs-v : (r : Rule) → A(Con-lhs-o r) → Var r  -- variables on the l.h.s.
          Con-rhs   : (r : Rule) → T O A (Var r)           -- a term on the r.h.s.
          Con-lab   : (r : Rule) → Label                   -- a label on the conclusion
```

Fig. 2. The data for a transition system specification, written in a dependently-typed *Agda*-like language

and a set X, we write $(\mathrm{T}\,O\,A\,X)$ for the set of all terms of the signature involving variables from X. (Thus T is a variation on the W-type construction of Martin-Löf type theory.)

Transition system specifications and GSOS. The schema for a positive GSOS transition system specification is readily translated into more type-theoretic notation — see Figure 2. The GSOS format has conditions about the appearance and disjointness of variables, which are not enforced by the type in Figure 2. They amount to two additional conditions:

- the map $(\mathit{Prem\text{-}lhs}\ r)$ factors through $(\mathit{Con\text{-}lhs\text{-}v}\ r)$; and
- the copairing $\langle \mathit{Prem\text{-}rhs}\ r, \mathit{Con\text{-}lhs\text{-}v}\ r\rangle : (\mathit{Prem}\ r + A(\mathit{Con\text{-}lhs\text{-}o}\ r)) \to \mathit{Var}\ r$ is an isomorphism.

A transition system specification of the type in Figure 2 gives rise to a labelled transition system over the ground terms of the signature:

$$(\longrightarrow) \subseteq (\mathrm{T}\ O\ A\ \emptyset) \times \mathit{Label} \times (\mathrm{T}\ O\ A\ \emptyset)$$

The usual definitions of bisimilarity and of congruence are appropriate here, and we have the following theorem:

Theorem 2. For the transition system induced by a specification in the positive GSOS format, bisimilarity is a congruence.

Generality. The above development is appropriate in *any* category that allows interpretation of type theory and quantifiers, and in particular, in any topos. (There is a technical caveat to Theorem 2, relating to the axiom of choice and 'internally projectivity'; details are in [10].)

The type in Figure 2 mentions a universe `Set` of sets, but this can be seen as shorthand. For example, a signature can be equivalently described as a function $s : Ar \to O$ between two sets; the arity of an operator $o \in O$ is its inverse image, the set $\{a \in Ar \mid s(a) = o\}$.

4 Theories of substitution

Nominal sets. Recall the theory of nominal sets, introduced by Pitts and Gabbay [7]. Fix an infinite set $\mathbb{A}$ of 'atoms'. A nominal set is a set X equipped with a permutation action, $\mathsf{Perm}(\mathbb{A}) \times X \to X$, satisfying a finite support requirement. A first example of a nominal set is the set of open value expressions $\mathbb{V}_{\mathrm{ex}}$, with variables in $\mathbb{A}$. The permutation action of this nominal set permutes the free variables of expressions; the support of a term is the set of variables that appear free in the term. As is usual, we write $a \ \# \ x$ to mean that a is not in the support of x. In our example, $\mathbb{V}_{\mathrm{ex}}$, this means that the expression x is equivalent to one in which the variable a does not appear.

I now introduce two theories of substitution over nominal sets, that arose in joint work with Marcelo Fiore; c.f. [4, Sec. 2.2].

Homogeneous substitution. We axiomatize the key properties of our set $\mathbb{V}$ of value expressions:

Definition 3. A *substitution algebra* is a nominal set $\mathbb{V}$ together with two equivariant functions, $\mathsf{i}_{\mathbb{V}} : \mathbb{A} \to \mathbb{V}$ and $\mathsf{salg}_{\mathbb{V}} : \mathbb{V} \times \mathbb{A} \times \mathbb{V} \to \mathbb{V}$, satisfying the following properties. We write $\{^{w}\!/_{a}\}v$ as shorthand for $\mathsf{salg}_{\mathbb{V}}(w, a, v)$, and elide $\mathsf{i}_{\mathbb{V}}$.

(i) If $a \ \# \ w$ then $a \ \# \ \{^{w}\!/_{a}\}v$.

(ii) If $b \ \# \ v$ then $\{^{b}\!/_{a}\}v = (a\ b) \cdot v$ (i.e. the action of the permutation $(a\,b)$, swapping a and b).

(iii) $\{^{v}\!/_{a}\}a = v$.

(iv) If $a \ \# \ u$ then $\{^{v}\!/_{a}\}u = u$.

(v) If $a \neq b$ and $a \ \# \ w$ then $\{^{w}\!/_{b}\}(\{^{v}\!/_{a}\}u) = \{^{(\{^{w}\!/_{b}\}v)}\!/_{a}\}(\{^{w}\!/_{b}\}u)$.

The set $\mathbb{V}_{\mathrm{ex}}$ of open value expressions, with free variables in $\mathbb{A}$, is a first example of a substitution algebra. Various authors have proposed ways to define substitution algebras by structural induction [3,5,6]. Our substitution algebras correspond with those of [3, Def. 3.1], when the carrier presheaves there preserve pullbacks of monos.

Heterogeneous substitution. A substitution action X allows substitution of values from $\mathbb{V}$ in elements of X.

Definition 4. Let $\mathbb{V}$ be a substitution algebra. A $\mathbb{V}$-*substitution action* is a nominal set X together with an equivariant function $\mathsf{sact}_X : \mathbb{V} \times \mathbb{A} \times X \to X$ that satisfies the following properties. We write $\{^{v}\!/_{a}\}x$ as shorthand for $\mathsf{sact}_X(v, a, x)$.

(i) If $a \ \# \ v$ then $a \ \# \ \{^{v}\!/_{a}\}x$.

(ii) If $b \ \# \ x$ then $\{^{b}\!/_{a}\}x = (a\ b) \cdot x$.

(iii) If $a \ \# \ x$ then $\{^{v}\!/_{a}\}x = x$.

(iv) If $a \neq b$ and $a \ \# \ w$ then $\{^{w}\!/_{b}\}(\{^{v}\!/_{a}\}x) = \{^{(\{^{w}\!/_{b}\}v)}\!/_{a}\}(\{^{w}\!/_{b}\}x)$

A *homomorphism* between $\mathbb{V}$-substitution actions, $X \to Y$, is given by an equivariant function $f : X \to Y$ that respects sact.

Just as substitution algebras correspond to a particular kind of monoid [3], substitution actions correspond to the monoid actions used in [5].

5 Value-passing systems are GSOS, categorically

We are now in a position to understand the specification of $\mathbb{V}$-CCS (fig. 1) as a GSOS specification. To formally specify the semantics, one must provide a structure of the type in Figure 2, not in terms of sets and functions, but instead in the category $\mathbf{SAct}(\mathbb{V})$ of $\mathbb{V}$-substitution actions and homomorphisms. The category $\mathbf{SAct}(\mathbb{V})$ is a topos, so it has enough structure to support the congruence theorem (thm. 2).

There is insufficient space here to write down the full specification structure for $\mathbb{V}$-CCS as a record of the type in Figure 2. It is illustrative to focus on the first two fields: the signature for the syntax of the calculus. For every value v in $\mathbb{V}$ there is an input prefix operator; it is not a unary operator, but it takes a variable and a term up-to α-equivalence. For any nominal set X one can construct a nominal set of α-equivalence classes:

$$[\mathbb{A}]X \;=\; (\mathbb{A}\times X)/_{\sim} \qquad \text{where if } b \mathrel{\#} x \text{ then } (a,x)\sim(b,(a\,b)\cdot x)$$

When X is a $\mathbb{V}$-substitution action, the nominal set $[\mathbb{A}]X$ coincides with the internal function space $[\mathbb{V}\to X]$ in $\mathbf{SAct}(\mathbb{V})$; the evaluation map $\mathbb{V}\times[\mathbb{A}]X\to X$ substitutes a value for the bound variable. So: as a binary operator is of arity 2, the binding input prefix operator is of arity $\mathbb{V}$. Thus fields O and A are defined. In this setting, $(\mathrm{T}\,O\,A\,\emptyset)$ is the set of $\mathbb{V}$-CCS terms, up-to α-equivalence, equipped with the appropriate capture-avoiding $\mathbb{V}$-substitution action.

Congruence of open bisimulation. In the category of $\mathbb{V}$-substitution actions, every relation is closed under substitution, and hence every bisimulation is open. We thus have the following corollary of Theorem 2:

Theorem 5. For any substitution algebra $\mathbb{V}$ containing the channel names, open bisimilarity is a congruence in $\mathbb{V}$-CCS.

Acknowledgements. Research funded by EPSRC grants GR/T22049/01 and EP/E042414/1. I appreciate feedback from referees, and I enjoyed discussing the material in this abstract with various people, most notably Marcelo Fiore. This work revisits the GSOS semantics for value-passing from [5], albeit in more direct terms.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, 2001.

[2] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[3] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding (extended abstract). In *LICS'99*, pages 193–202, 1999.

[4] M. P. Fiore and S. Staton. A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In *LICS'06*, pages 49–58, 2006.

[5] M. P. Fiore and D. Turi. Semantics of name and value passing (extended abstract). In *LICS'01*, pages 93–104, 2001.

[6] M. J. Gabbay and A. Mathijssen. Capture-avoiding substitution as a nominal algebra. In *ICTAC 2006*, volume 4281 of *Lecture Notes in Comput. Sci.*, pages 198–212. Springer, 2006.

[7] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[8] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[9] D. Sangiorgi. A theory of bisimulation for the pi-calculus. *Acta Inform.*, 33(1):69–97, 1996.

[10] S. Staton. General structural operational semantics through categorical logic. In *LICS'08*, pages 166–177, 2008.