Factorization in Process Domains (Second Approach)

Tom Verhoeff

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB EINDHOVEN, The Netherlands
E-mail: wstomv@win.tue.nl

March 1991, Revised June 1995

In this note we present a general theory about factorization in process domains.

1 Introduction

By way of introduction, consider a process domain P together with a binary relation $\sqsubseteq$ (refinement relation: $p \sqsubseteq q$ expresses 'q refines p'[1]) and a binary operator $\parallel$ (parallel composition). We assume that $\langle P, \sqsubseteq \rangle$ is a complete poset, in the sense that every subset has a greatest lower bound. Furthermore, we assume that $\parallel$ has unit e and is commutative, associative, and (universally) $\sqcap$-junctive. The latter means that for process q and set of processes V we have

$$(\sqcap V) \parallel q \quad = \quad (\sqcap p : p \in V : p \parallel q). \tag{1}$$

Finally, we assume that refinement is **fully abstract** with respect to a form of **testing**, that is, there exists a process d such that

$$p \sqsubseteq q \quad \equiv \quad (\forall r : p \parallel r \sqsupseteq d : q \parallel r \sqsupseteq d). \tag{2}$$

One may interpret

$$p \parallel r \quad \sqsupseteq \quad d$$

as 'process p passes the test in the context of process r'. An immediate consequence of (2) is

$$p = q \quad \equiv \quad (\forall r :: p \parallel r \sqsupseteq d \equiv q \parallel r \sqsupseteq d), \tag{3}$$

that is, two processes are equal if and only if they pass the same tests. It the implication from right to left that is important for full abstraction.

We are interested in the **design equation**

$$p :: p \parallel q \quad \sqsupseteq \quad r \tag{4}$$

for given processes q and r. The equation arises when one has decided to implement specification r as the parallel composition of process q with some yet unknown process p. In particular, we would like to know the $\sqsubseteq$-least solution to the design equation, which—if it exists—could serve as specification for p.

2 Theory

We define the **reflection** $\backsim q$ of process q by

$$\backsim q \quad = \quad (\sqcap p : p \parallel q \sqsupseteq d : p). \tag{5}$$

It is a proper definition because all glb's exist. Notationally, reflection binds stronger than parallel composition. According to Corollary 2.3 below, the reflection of q is the $\sqsubseteq$-least solution to a very particular design equation, viz. $p :: p \parallel q \sqsupseteq d$. The reflection of q can

[1] Often, $p \sqsubseteq q$ corresponds to 'q is more deterministic than p'.

also be interpreted as the severest test that q passes. In Corollary 2.9 we shall see that the general design equation can be solved in terms of reflection and parallel composition.

First of all we show that $\backsim q$ is a solution to the design equation $p :: p \parallel q \sqsupseteq d$.

2.1 Property For process q we have

$$\backsim q \parallel q \quad \sqsupseteq \quad d.$$

Proof We derive for process q

$$\begin{array}{ll} & \backsim q \parallel q \\ = & \quad \{ \text{ definition of reflection: (2) } \} \\ & (\sqcap\, p : p \parallel q \sqsupseteq d : p) \parallel q \\ \equiv & \quad \{ \text{ parallel composition is } \sqcap\text{-junctive: (1) } \} \\ & (\sqcap\, p : p \parallel q \sqsupseteq d : p \parallel q) \\ \sqsupseteq & \quad \{ \text{ property of greatest lower bound } \} \\ & d \end{array}$$

■

The next property turns out to be a very useful characterization of reflection. In the remainder of this note, it will be used instead of the definition of reflection.

2.2 Property For processes p and q we have

$$p \parallel q \sqsupseteq d \quad \equiv \quad p \sqsupseteq \backsim q.$$

Proof We derive the implication from left to right

$$\begin{array}{ll} & p \parallel q \sqsupseteq d \\ \Rightarrow & \quad \{ \text{ property of greatest lower bound } \} \\ & p \sqsupseteq (\sqcap\, p : p \parallel q \sqsupseteq d : p) \\ \equiv & \quad \{ \text{ definition of reflection: (5) } \} \\ & p \sqsupseteq \backsim q \end{array}$$

and the implication from right to left

$$\begin{array}{ll} & p \sqsupseteq \backsim q \\ \equiv & \quad \{ \text{ full abstraction of refinement: (2) } \} \\ & (\forall\, r : \backsim q \parallel r \sqsupseteq d : p \parallel r \sqsupseteq d) \\ \Rightarrow & \quad \{ \text{ instantiation with } r := q, \text{ using Property 2.1 } \} \\ & p \parallel q \sqsupseteq d \end{array}$$

■

2.3 Corollary The set of solutions to the design equation $p :: p \parallel q \sqsupseteq d$ is $\sqsubseteq$-upward closed and the $\sqsubseteq$-least solution is given by $\backsim q$. ■

It turns out that d is the reflection of the unit e, irrespective of which process d actually is. Note, however, that the choice of d is limited by (2) and that reflection depends on d.

2.4 Property We have

$$\backsim e \quad = \quad d.$$

Proof We derive for arbitrary process r

$$\begin{array}{ll} & r \sqsupseteq \backsim e \\ = & \quad \{ \text{ Property 2.2 } \} \\ & r \parallel e \sqsupseteq d \\ = & \quad \{ e \text{ is unit of parallel composition } \} \\ & r \sqsupseteq d \end{array}$$

On account of

$$p = q \quad \equiv \quad (\forall\, r :: r \sqsupseteq p \equiv r \sqsupseteq q), \tag{6}$$

we now infer $\backsim e = d$. ■

Reflection reverses the order.

2.5 Property For processes p and q we have

$$p \sqsubseteq q \quad \equiv \quad \backsim p \sqsupseteq \backsim q.$$

Proof For processes p and q we derive

$$\begin{array}{ll} & p \sqsubseteq q \\ \equiv & \quad \{ \text{ refinement is fully abstract: (2) } \} \\ & (\forall\, r : p \parallel r \sqsupseteq d : q \parallel r \sqsupseteq d) \\ \equiv & \quad \{ \text{ parallel composition is commutative } \} \\ & (\forall\, r : r \parallel p \sqsupseteq d : r \parallel q \sqsupseteq d) \\ \equiv & \quad \{ \text{ Property 2.2 } \} \\ & (\forall\, r : r \sqsupseteq \backsim p : r \sqsupseteq \backsim q) \\ \equiv & \quad \{ \text{ property of partial ordering } \} \end{array}$$

$$\backsim p \sqsupseteq \backsim q$$

■

The next property is a slight modification of Property 2.2.

2.6 Property For processes p and q we have

$$p \sqsupseteq q \quad \equiv \quad p \parallel \backsim q \sqsupseteq d.$$

Proof We derive for processes p and q

$$\begin{array}{ll} & p \sqsupseteq q \\ \equiv & \{ \text{ Property 2.5 } \} \\ & \backsim q \sqsupseteq \backsim p \\ \equiv & \{ \text{ Property 2.2, using commutativity of } \parallel \} \\ & p \parallel \backsim q \sqsupseteq d \end{array}$$

■

Reflection is an involution, that is, its own inverse.

2.7 Property For process p we have

$$\backsim\backsim p \quad = \quad p.$$

Proof We derive for processes p and r

$$\begin{array}{ll} & r \sqsupseteq \backsim\backsim p \\ \equiv & \{ \text{ Property 2.2 } \} \\ & r \parallel \backsim p \sqsupseteq d \\ \equiv & \{ \text{ Property 2.6 } \} \\ & r \sqsupseteq p \end{array}$$

On account of (6), we now infer $\backsim\backsim p = p$. ■

Finally, we can derive an elegant expression for the $\sqsubseteq$-least solution to the design equation (4).

2.8 Theorem For processes p, q, and r we have

$$p \parallel q \sqsupseteq r \quad \equiv \quad p \sqsupseteq \backsim(q \parallel \backsim r).$$

Proof For processes p, q, and r we derive

$$\begin{array}{ll} & p \parallel q \sqsupseteq r \\ \equiv & \{ \text{ Property 2.6 } \} \\ & (p \parallel q) \parallel \backsim r \sqsupseteq d \\ \equiv & \{ \text{ parallel composition is associative } \} \\ & p \parallel (q \parallel \backsim r) \sqsupseteq d \\ \equiv & \{ \text{ Property 2.2 } \} \\ & p \sqsupseteq \backsim(q \parallel \backsim r) \end{array}$$

■

2.9 Corollary The $\sqsubseteq$-least solution to the design equation (4) is

$$\backsim(q \parallel \backsim r). \tag{7}$$

■

A common notation for the $\sqsubseteq$-least solution to the design equation (4) is r/q. The operator $/$ on processes is completely characterized by the equivalence

$$p \parallel q \sqsupseteq r \quad \equiv \quad p \sqsupseteq r/q. \tag{8}$$

■ Using this notation, reflection can be expressed by $\backsim q = d/q$. It is often more convenient to manipulate $/$ using (8) than to manipulate the fairly awkard definition of $/$ in terms of $\backsim$ and $\parallel$.

3 Additional Results

Note that in the preceding section we have used the $\sqcap$-junctivity of parallel composition only once, viz. in Property 2.1, where it is used to distribute parallel composition over the greatest lower bound of a very particular set of processes. Hence, to obtain all results, in particular Theorem 2.8, it suffices to assume Property 2.1 instead of the $\sqcap$-junctivity of $\parallel$. But $\sqcap$-junctivity of $\parallel$ cannot be denied, because it follows from Theorem 2.8:

Proof Let q be a process and V a set of processes. We derive for process r

$$\begin{array}{ll} & (\sqcap V) \parallel q \sqsupseteq r \\ \equiv & \{ \text{ Theorem 2.8 } \} \\ & \sqcap V \sqsupseteq \backsim(q \parallel \backsim r) \\ \equiv & \{ \text{ property of greatest lower bound } \} \\ & (\forall\, p : p \in V : p \sqsupseteq \backsim(q \parallel \backsim r)) \\ \equiv & \{ \text{ Theorem 2.8 } \} \\ & (\forall\, p : p \in V : p \parallel q \sqsupseteq r) \end{array}$$

$\equiv$ { property of greatest lower bound }
$(\sqcap p : p \in V : p \parallel q) \sqsupseteq r$

On account of (6), we then have

$$p \parallel \sqcap V \quad = \quad (\sqcap q : q \in V : p \parallel q). \tag{9}$$

∎

Finally, we observe that, since $\langle P, \sqsubseteq \rangle$ is self-dual (under $\backsim$), it is not only a complete poset but, in fact, a complete lattice, in the sense that every subset has a greatest lower bound and a least upper bound. Actually, the completeness assumption is too strong. We only used the existence of greatest lower bounds of sets like

$$\{r : p \parallel r \sqsupseteq d : r\},$$

for instance, in the definition of reflection.

4 Conclusion

We have shown that under certain conditions (especially full abstraction: (2)) the design equation over a process domain has an elegant solution in terms of a reflection operator. In various domains, the reflection operator is very simple. Of course, our presentation raises new questions. These will be addressed elsewhere.

I would like to thank Roland Backhouse for some helpful comments on an earlier version. In particular his insistence on ‘narrower’ proofs has stimulated me.