On Testing the Observable Actions of Processes

William Ferreira

Abstract. We present and investigate two testing preorders for a value-passing version of CCS, [Mil89] which distinguish processes by their observable actions. We develop an operational theory for the preorders, and compare

where $\|$ is the parallel operator, τ is the action resulting from an internal computation or communication, and ω is an observer success state. However we have that:

characterisations for them, defined independently of contexts. In section 4 we review *must* testing for *VPL*, and then compare *must* testing to *guarantee* and *strongly guarantee* testing. We prove an expressivity result relating *must* and *guarantee* testing under an assumption about the operational semantics of the conditional expression $\mathsf{if}\ \cdot\ \mathsf{then}\ \cdot\ \mathsf{else}\ \cdot$. In section 5 we construct two denotational models for the language, based on variations of *value-passing acceptance trees* [HI93], and in section 6 we prove that these models are fully abstract for their respective preorders.

2 Operational Semantics

In this section we present the syntax and operational semantics of *VPL*, the value-passing version of τ-less CCS introduced in [HI93]. Let:

- $v, v_1, v_2, \ldots \in \mathit{Val}$ be a set of values,
- $x, x_1, x_2, \ldots \in \mathit{Var}$ a set of expression variables,
- $op \in \mathit{Op}$ a set of functions or operator symbols,
- $X, Y, Z \in \mathit{VRec}$ a set of process variables, and
- $n, n_1, n_2, \ldots \in \mathit{Chan}$ a predefined set of channel names.

The abstract syntax of our language is given by the following grammar:

$$
\begin{aligned}
e, e_1, \ldots \in \mathit{Exp} &:= \mathbf{0} \mid \alpha.e \mid \mathsf{if}\, l\, \mathsf{then}\, e\, \mathsf{else}\, e \mid e \,\square\, e \mid e\backslash n \mid e[R] \mid \mu X.e \mid X \mid \Omega \\
\square \in \mathit{BinOp} &:= \oplus \mid + \mid \| \\
\alpha, \alpha_1, \alpha_2, \ldots \in \mathit{Pre} &:= n?x \mid n!l \\
l, l_1, l_2, \ldots \in \mathit{SExp} &:= \mathsf{true} \mid \mathsf{false} \mid op(\vec{l_i}) \mid v \mid x
\end{aligned}
$$

The set *Val* could be any flat domain of values such as the integers, in which *Op* would consist of the familiar operations of addition, subtraction etc.; we also assume that *Op* includes the Boolean operators. We ignore types, and assume that for any expression $\mathsf{if}\, l\, \mathsf{then}\, e_1\, \mathsf{else}\, e_2$ that l is a Boolean-valued expression, and that the use of the operator symbols *op* is type-respecting. We use the standard definition of free and bound variables for expressions, and use $\mathit{free}(e)$ to denote the set of free expression variables in e. We use $e\{\vec{v_i}/\vec{x_i}\}$ for the simultaneous substitution of values $\vec{v_i}$ for free expression variables $\vec{x_i}$ in e, while $e[\vec{e_i}/\vec{X_i}]$ denotes the simultaneous substitution of terms $\vec{e_i}$ for free process variables $\vec{X_i}$ in e. We use *VPL* to denote the set of closed expressions in *Exp*, which we refer to as *processes*. The constructs of *VPL* have the following informal meaning:

- $\mathsf{if}\, l\, \mathsf{then}\, e_1\, \mathsf{else}\, e_2$ – a process that behaves like e_1 if l evaluates to true, and like e_2 otherwise,
- $\alpha.e$ – a process that performs the communication action specified by α and then behaves like e,
- $e_1 \oplus e_2$ – a process that can evolve to either e_1 or e_2 without interaction with the environment,
- $e_1 + e_2$ – a process that behaves like e_1 or e_2 depending on the behaviour of the environment,
- $e_1 \| e_2$ – a process that allows the interleaving of the behaviours of e_1 and e_2, or communication between them,
- $e\backslash n$ – a process that behaves like e except that it cannot offer communications actions on channel n to the environment,
- $\mathbf{0}$ – the inactive process,
- $\mu X.e$ – the recursive process,
- $e[R]$ – a process that behaves like e except that the channel names of actions performed by e are renamed according to the renaming function R and,
- Ω – the undefined or divergent process

We now present the operational semantics for processes, and to make things simpler we ignore the evaluation of Boolean expressions. That is we assume that for each closed Boolean simple expression l there is a corresponding truth value $[\![l]\!]$ and more generally for any Boolean simple expression l

(Bot) $\dfrac{}{\Omega \xrightarrow{\tau} \Omega}$ (Fix) $\mu X.e$

For both preorders, we define their kernels $\eqsim_{\mathcal{G}}$ and $\eqsim_{\mathcal{SG}}$ as $\sqsubseteq\!\!\!\!\sim_{\mathcal{G}} \cap \sqsubseteq\!\!\!\!\sim_{\mathcal{G}}{}^{-1}$ and $\sqsubseteq\!\!\!\!\sim_{\mathcal{SG}} \cap \sqsubseteq\!\!\!\!\sim_{\mathcal{SG}}{}^{-1}$ respectively.

The universal quantification over contexts in the definitions of the *guarantee* and *strong guarantee* preorders

2. $\mathcal{A}(e_2, s) \ll \mathcal{A}$

$$
\begin{aligned}
&\neq \{\{m!\}, \{m'!\}\} \\
&= \mathcal{A}(e_2, \varepsilon)
\end{aligned}
$$

but:

$$\mathcal{A}$$

PROOF. For the *if* case we prove the contra-positive, so suppose that $e \not\Downarrow^{\mathcal{G}} s$. If $s = \varepsilon$ then we have $e \Uparrow$ in which case $\mathbb{N}_s[e] \Uparrow$ as well; otherwise for some s_1, s_2 with $s = s_1 s_2$ we have either:

- $e \stackrel{s_1}{\Longrightarrow} e', e' \Uparrow$ – in this case we can show that $\mathbb{N}_s[e] \stackrel{\varepsilon}{\Longrightarrow} e_1$ with either:

  $$e_1 = e' \parallel \textsf{if true then } con(s_2) \textsf{ else } \mathbf{0}$$

  or,

  $$e_1 = e' \parallel con(s')$$

  and therefore $\mathbb{N}_s[e] \Uparrow$ or,

- $s_1 = s'_1.n!v$ and $e \xrightarrow{s'_1.n!v'} e'$ with $e' \Uparrow$ – in this case we can show that:

  $$\mathbb{N}_s[e] \stackrel{\varepsilon}{\Longrightarrow} e' \parallel \textsf{if false then } con(s'') \textsf{ else } \mathbf{0}$$

  and therefore $\mathbb{N}_s[e] \Uparrow$.

The *only if* case is proved by induction on s. If $s = \varepsilon$ then $\mathbb{N}_s[e] = e \parallel \mathbf{0}$ and therefore $\mathbb{N}_s[e] \Downarrow$ implies $e \Downarrow$

PROOF. Assume the hypotheses of the proposition are true; firstly we show that:

$$e \overset{\varepsilon}{\Longrightarrow} e' \text{ if and only if } \mathbb{T}^{n,f}_{s,A}[e] \overset{f(s)}{\Longrightarrow} \mathbb{T}^{n,f}_{\varepsilon,A}[e']$$

by induction on s. For the *only if* part of the proposition we prove the contra-positive, so suppose there exists $B \in \mathcal{A}(e,s)$ such that $A \cap B = \emptyset$. Therefore $e \overset{s}{\Longrightarrow} e' \overset{\tau}{\not\longrightarrow}$ and $B = \mathcal{S}(e')$ for some e'. By examination of the transitions from $\mathbb{T}^{n,f}_{s,A}[e]$ we can show that:

$$\mathbb{T}^{n,f}_{s,A}[e] \overset{f(s)}{\Longrightarrow} (e' \parallel \mathbf{0})[R^A_n]$$

and therefore:

$$\mathbb{G}^{n,f}_{s,A}[e] \overset{\varepsilon}{\Longrightarrow} (e' \parallel \mathbf{0})[R^A_n] \parallel \mathbf{0}$$

Since $e' \overset{a}{\not\longrightarrow}$ for any $a \in A$ we have that $\mathbb{G}^{n,f}_{s,A}[e] \not\Downarrow^{\mathcal{G}} n!$. For the *only if* case the proof is by induction on s. □

The class of contexts needed to characterise *strong acceptances* is similar to that for the acceptances, except we need to record some additional information in the context about the set of prefixes A. Let $In(A)$ denote the elements of A which are input prefixes, i.e. of the form $n?$ for some n, and f_A a finite partial function from $In(A)$ to Val. We define the context $\mathbb{S}^{n,f}_{s,A}$ by:

$$\mathbb{S}^{n,f}_{s,A} \overset{\text{def}}{=} [\,] \parallel \mathit{strong}(s,A,f,n)$$

where:

$$\begin{aligned}
\mathit{strong}(\varepsilon,A,f,n) &\overset{\text{def}}{=} \mathit{strong}(A,f,n)\\
\mathit{strong}(n?v.s,A,f,n) &\overset{\text{def}}{=} n!v.\mathit{strong}(s,A,f,n) + n!\\
\mathit{strong}(n!v.s,A,f,n) &\overset{\text{def}}{=} n?x.\mathsf{if}\ x = v\ \mathsf{then}\ \mathit{strong}(s,A,f,n)\ \mathsf{else}\ n! + n!
\end{aligned}$$

and:

$$\begin{aligned}
\mathit{strong}(A,f,n) &\overset{\text{def}}{=} \sum\{\mathit{strong}(\pi,f) \mid \pi \in A\}\\
\mathit{strong}(n?,f) &\overset{\text{def}}{=} n!f(n?).n!\\
\mathit{strong}(n!,f) &\overset{\text{def}}{=} n?x.n!
\end{aligned}$$

The set of prefixes A in the context $\mathbb{S}^{n,f}_{s,A}$ represent prefixes drawn from the strong acceptances of a process e after some sequence of actions s has been performed. In this case, by the definition of strong acceptances,strong acceptance45.52031TdΩ(=)acceptances, THEOREM 3.19. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \underset{\sim}{\sqsubset}_{\mathcal{G}} e_2 \textit{ implies } e_1 \ll_{\mathcal{G}} e_2$$

PROOF. We prove the

Therefore if $e_1 \sqsubseteq_{\mathcal{MT}} e_2$ and $\mathbb{C}[e$

LEMMA 4.2. *Let $O \in \mathcal{O}^+$ be an open term with free variables $\vec{x}$, and ρ a substitution with $\vec{x_i} \subseteq dom(\rho)$, then:*

$$O\rho \xrightarrow{\omega} \text{ implies } O\rho' \xrightarrow{\omega}$$

for all substitutions with $\vec{x_i} \subseteq dom(\rho')$.

PROOF. The proof is by induction on the structure of O. □

The import of this lemma is that there are many more observers in $\mathcal{O}^+$ which are capable of performing the success action ω. This is precisely what makes $\sqsubseteq^+_{\mathcal{MT}}$ no more discriminating than $\sqsubseteq^+_{\mathcal{G}}$ for VPL^+.

THEOREM 4.3. *For $e_1, e_2 \in VPL^+$ we have:*

$$e_1 \sqsubseteq^+_{\mathcal{MT}} e_2 \text{ if and only if } e_1 \sqsubseteq^+_{\mathcal{G}} e_2$$

PROOF. The proof of the *only if* case uses the fact that for observers:

$$O_? \stackrel{\text{def}}{=} n!v.\omega \oplus n!v.\omega \text{ and,}$$
$$O_! \stackrel{\text{def}}{=} n?x.\omega \oplus n?x.\omega$$

$O_? \;||$

Another interesting property of $\sqsubseteq_{\mathcal{G}}$ is that its discriminatory power is dependent on the presence of the renaming operator; this is implicit in the proof of PROPOSITION 3.17. For example suppose that the operator $[R]$ is removed from the language, then we have no way of distinguishing between the two terms:

$$e_1 \stackrel{\text{def}}{=} n_1!v.((n_2!v.\Omega + n_3!v.\Omega) \oplus (n_4!v.\Omega + n_5!v.\Omega)) \text{ and: } e_2 \stackrel{\text{def}}{=} n_1!v.(n_3!v.\Omega \oplus n_5!v.\Omega)$$

First note that we cannot use any of the prefixes $n_2 \dots n_5$ to distinguish between e_1 and e_2 because there is no context $\mathbb{C}$ and $n_i!$ for $2 \leq i \leq 5$ such that $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} n_i!$. If we try to utilise some fresh prefix π, then we run into problems because any context that tries to communicate with sub-terms $(n_2!v.\Omega + n_3!v.\Omega)$ or $(n_4!v.\Omega + n_5!v.\Omega)$ of e_1 to guarantee π, will leave e_1 in a divergent state. The renaming operator allows the context to avoid making any communication, by renaming the actions of the process that we wish to communicate with to some fresh action. Note that e_1 and e_2 are distinguished in $\sqsubseteq_{\mathcal{G}}$ by the context:

$$\mathbb{C} \stackrel{\text{def}}{=} ([\,] \parallel n?x.\mathbf{0})[R]$$

where:

$$R(n_i) = \begin{cases} n' & \text{if } i = 2,4 \\ n_i & \text{otherwise} \end{cases}$$

where n' is a fresh channel name, since $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} n'!$ and $\mathbb{C}[e_2] \not\downarrow^{\mathcal{G}} n'!$.
To recapture the testing power of $\sqsubseteq_{\mathcal{G}}$ without the renaming operator we need to strengthen the predicate $\cdot \downarrow^{\mathcal{G}} \cdot$ to sets of prefixes, i.e. we need to define $\cdot \downarrow^{\mathcal{G}} \cdot$ as:

$$e \downarrow^{\mathcal{G}} A \text{ if } e \Downarrow \text{ and } e \stackrel{\varepsilon}{\Longrightarrow} e' \text{ implies } e' \stackrel{\pi}{\Longrightarrow} \text{ for some } \pi \in A$$

Let $\sqsubseteq$ be the preorder derived from the above definition of $\cdot \downarrow^{\mathcal{G}} \cdot$ by closing up under all contexts. Then we have $e_1 \not\sqsubseteq e_2$ since $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} \{n_1!, n_3!\}$ and $\mathbb{C}[e_2] \not\downarrow^{\mathcal{G}} \{n_1!, n_3!\}$ where:

$$\mathbb{C} \stackrel{\text{def}}{=} ([\,] \parallel n?x.\mathbf{0})$$

We have the following result:

PROPOSITION 4.4. *For* $e_1^{\text{def5()2()150d(ee)453020d(e)15+12024f331}}$

An *interpretation* of *VPL* in a domain D is given by a *semantic function* $D[\![\ ]\!]$ with type:

$$D[\![\ ]\!] : Exp \longrightarrow [Env_V \longrightarrow [Env_D \longrightarrow D]]$$

where Env_V denotes the set of *Val* environments: mappings from the set of variables *Var* to the set of values *Val*, and Env_D is the set of D environments: mappings from the set of process variables *VRec* to the model D. The function $D[\![\ ]\!]$ is defined by structural induction on expressions as:

$$\begin{aligned}
D[\![x]\!]\rho\sigma &= \rho(x)\\
D[\![\mathbf{0}]\!]\rho\sigma &= \mathbf{0}_D\\
D[\![\Omega]\!]\rho\sigma &= \bot\\
D[\![e[R]]\!]\rho\sigma &= rename_D\ R\,[\![e]\!]\rho\sigma\\
D[\![op(\vec{l_i})]\!]\rho\sigma &= [\![op]\!](\rho(\vec{l_i}))\ \text{ for each } op \in Op\\
D[\![\Box(\vec{e_i})]\!]\rho\sigma &= \Box_D(D[\![\vec{e_i}]\!]\rho\sigma) \quad \text{for } \Box \in \{\oplus, +, \|\}\\
D[\![\mu X.e]\!]\rho\sigma &= \mathsf{fix}(\lambda d.D[\![e]\!]\rho\sigma[X \mapsto d])\\
D[\![\mathsf{if}\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2]\!]\rho\sigma &= \begin{cases} D[\![e_1]\!]\rho & \text{if } [\![l]\!]\rho\sigma = \mathsf{true}\\ D[\![e_2]\!]\rho & \text{otherwise}\end{cases}\\
D[\![n?x.e]\!]\rho\sigma &= in_D\ n\ \lambda v.D[\![e]\!]\rho[x \mapsto v]\sigma\\
D[\![n!l.e]\!]\rho\sigma &= \begin{cases} out_D\ n\ \rho(l)\ D[\![e]\!]\rho\sigma & \text{if } l \in Var\\ out_D\ n\ l\ D[\![e]\!]\rho\sigma & \text{otherwise}\end{cases}
\end{aligned}$$

where each of the functions $\Box_D, rename_D$ are continuous on D, and the functions in_D and out_D have type:

$$\begin{aligned}
in_D &: Chan \longrightarrow ((Val \longrightarrow D) \longrightarrow D)\\
out_D &: Chan \longrightarrow (Val \longrightarrow (D \longrightarrow D))
\end{aligned}$$

where in_D is continuous in its second argument and out_D is continuous in its third argument, and fix is the least fixed point operator. In [Ing94] an interpretation for *VPL* is given in domain AT^v.

The goal of the next section is to show how models $\mathbf{G}$ and $\mathbf{SG}$ are *fully abstract* with respect to the preorders $\sqsubset$

$$
\begin{aligned}
&= v_1 \otimes \bot \\
&= v_2 \otimes \bot \\
&= \bot
\end{aligned}
$$

and using $(Val \otimes D)$ as the domain for modelling the sequels to output prefixes. Suppose D is a domain and $\oplus_D$ is a continuous function on D satisfying for all elements $d_1, d_2 \in D$:

$$d_1 \oplus_D d_2 \leq d_1 \tag{1}$$

$$d_1 \oplus_D d_2 = d_2 \oplus_D d_1 \tag{2}$$

$$d \oplus_D d = d \tag{3}$$

then the pair $\langle D, \oplus_D \rangle$ is called a *continuous upper semi-lattice* [Gun92, Hen94]. We will use the function $\oplus_D$ as the interpretation of the internal choice operator $\oplus$ of *VPL*. We sometimes write $\langle D, \oplus_D \rangle$ for the domain D with a continuous function $\oplus_D$ satisfying (1) – (3) above.

Suppose $\langle D, \oplus_D \rangle$ and $\langle E, \oplus_E \rangle$ are domains:

- $f : Val \times D \longrightarrow Val \times E$ is *right-linear* if for elements $d_1, d_2 \in D$:

$$f(v, d_1) \oplus_E f(v, d_2) = f(v, d_1 \oplus_D d_2)$$

- $f : D \longrightarrow E$ is *linear* if for $d_1, d_2 \in D$:

$$g(d_1 \oplus_D d_2) = g(d_1) \oplus_E g(d_2) \text{ and,}$$

- $f : Val \times D \longrightarrow E$ is *right-strict* if:

$$f(v, \bot_D) = \bot_E$$

For domain $\langle D, \oplus_D \rangle$ let $(Val \otimes D)$ be the set characterised by the following universal property:

1. there isDiGr,990TdΩ424TfΩ8.40407(60160Td23917.7-1.43984TdΩ[(sat)TjΩR1260.ΩR12654(har–R1260.24T

where:

$$G \stackrel{\text{def}}{=} \mathsf{fix}(\lambda F.\lambda Z.(Z \cup F(\{(v, k_1 \oplus_D k_2) \mid \{(v, k_2), (v, k_2)\} \in Z\})))$$

We will write $\oplus_\otimes$ to refer to $\oplus_{Val \otimes D}$.

PROPOSITION 5.1. *$\langle Val \otimes D, \oplus_\otimes \rangle$ satisfies the universal property given above.*

PROOF. Let $\imath$

Let $F_{\mathbf{SG}}$

by taking the contexts:

$$\mathbb{C}_1 \stackrel{\text{def}}{=} [\,] \parallel n!2.m!.\mathbf{0} \text{ and,}$$
$$\mathbb{C}_2 \stackrel{\text{def}}{=} [\,] \parallel n!1.m!.\mathbf{0}$$

since $\mathbb{C}_1[e_1] \downarrow^{\mathcal{SG}} m!, \mathbb{C}_2[e_2] \downarrow^{\mathcal{SG}} m!$ and obviously $\mathbb{C}_i[\mathbf{0}] \not\downarrow^{\mathcal{SG}} m!$. When e_1 and e_2 are combined using $\oplus$ the prefix $n?$ becomes a divergence of the process $e_1 \oplus e_2$, although it is not a divergence of either e_1 or e_2. Let f_{even} and f_{odd} be the functions which converge for even and odd values respectively, and diverge otherwise. From the definition of $\oplus_{\mathbf{SG}}$ we have:

SG

Then the term:

$$\bigoplus\{e_A \mid A \in \mathcal{A}\}$$

is in *head normal form* if each e_A is the simple sum form:

$$\sum\{e_a \mid a \in A\}$$

where $\bigoplus$ denotes the application of the operator $\oplus$ to a non-empty, finite set of expressions, and $\sum$ the application of $+$ to a finite set of expressions, where by convention if the set is finite then the expression denotes $\mathbf{0}$. Let $\cdot \xrightarrow{a}_{AT^v} \cdot$

- $s = n?v.s'$ – this case is simpler than the case $s = n!v.s'$.

□

LEMMA 6.6.

and for each $d \in \mathbf{SG}$ and $s \in Act^*$ let $\mathcal{A}_{\mathcal{S}}(d, s)$ denote the obvious extension of the acceptances of d after s to elements of $\mathbf{SG}$.

DEFINITION 6.8. For $d_1, d_2 \in \mathbf{SG}$ and $s \in Act^*$ let $d_1 \ll_{\mathbf{SG}} d_2$ if $d_1 \Downarrow s$ implies:

- $d_2 \Downarrow s$,
- $\mathcal{D}(d_2, s) \subseteq \mathcal{D}(d_1, s)$ and,
- $\mathcal{A}_{\mathcal{S}}(d_2, s) \subseteq \mathcal{A}_{\mathcal{S}}(d_1, s)$.

□

We have the following result which is the analogue for $\mathbf{SG}$ of THEOREM 6.3:

PROOF. Since $e \Downarrow^{\mathcal{G}} s$ we have $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{SG}[\![e]\!] = \mathbf{SG}[\![hnf(e)]\!]$, so it is sufficient to show that:

$$c_{\mathcal{D}(hnf(e),s)}(\mathcal{A}_{\mathcal{S}}(hnf(e), s)) = \mathcal{A}_{\mathcal{S}}(\mathbf{SG}[\![hnf(e)]\!], s)$$

The proof is by induction on s, and follows from LEMMA 6.10, the structure of head normal forms and the interpretations of the operators $\oplus, +$ and $\alpha.$ in $\mathbf{SG}$. □

We can now present our final result:

THEOREM 6.14. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \sqsubseteq_{\mathcal{SG}} e_2 \text{ if and only if}$$

[Plo81b] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Dept, Aarhus University, Denmark, 1981.

[San92] Davide Sangiorgi. *Expresing Mobility in Process Algebras: First Order and Higher-Order Paradigms*. Ph.D. thesis, LFCS, Edinburgh University, 1992.

A Interpretation of the remaining operators of *VPL* in G and SG.

Let $rename_{\mathbf{G}}$ be defined by $rename_{\mathbf{G}} \stackrel{\text{def}}{=} \lambda R.\,\mathsf{fix}(\lambda F.\mathsf{up}(\Theta_R\, F))$ where:

$$\Theta_R\, F\, \langle \mathcal{A}, f\rangle \stackrel{\text{def}}{=} \bigoplus_{\mathbf{G}} \{ \sum_{\mathbf{G}} T_A \mid A \in \mathcal{A}\}$$

and:

$$T_A \stackrel{\text{def}}{=} \{ in_{\mathbf{G}}\ R(n)\ \lambda v.F(f(n?)(v)) \mid n? \in A\} \ \cup\ \{ out_{\mathbf{G}}\ R(n)\ v\ F(d) \mid n! \in A, \{(v,d)\} \subseteq f(n!)\}$$

and where we have used $\bigoplus_{\mathbf{G}}$ and $\sum_{\mathbf{G}}$ to denote the application of $\oplus_{\mathbf{G}}$ and $+_{\mathbf{G}}$ to finite subsets of $\mathbf{G}$.
Let $+_{\mathbf{SG}}$ be the strict extension of the following function:

$$\langle \mathcal{A}, X, f\rangle +_{\mathbf{SG}} \langle \mathcal{B}, Y, g\rangle \stackrel{\text{def}}{=} \langle c_Z(\mathcal{A} \vee \mathcal{B}), Z, (f_{in} \oplus_{\mathbf{SG}} g_{in})\lceil \quad \uplus (f_{out} \oplus_{\mathbf{SG}} g_{out})\lceil \quad \rangle$$

where:

$$\stackrel{\text{def}}{=} (|\mathcal{A}| \cup |\mathcal{B}|) \setminus Z,$$

$$Z \stackrel{\text{def}}{=} X \cup Y \cup \Omega(f, g) \text{ and,}$$

$$\mathcal{A} \vee \mathcal{B} \stackrel{\text{def}}{=} \{A \cup B \mid A \in \mathcal{A}, B \in \mathcal{B}\}$$