A Lattice-Theoretic Model for an Algebra of Communicating Sequential Processes

Malcolm Tyrrell[1,3], Joseph M. Morris[1,3],
Andrew Butterfield[2,3], and Arthur Hughes[2]

[1] School of Computing, Dublin City University, Dublin 9, Ireland
{Malcolm.Tyrrell, Joseph.Morris}@computing.dcu.ie
[2] Department of Computer Science, Trinity College, Dublin 2, Ireland
{Andrew.Butterfield, Arthur.Hughes}@cs.tcd.ie
[3] Lero, the Irish Software Engineering Research Centre

Abstract. We present a new lattice-theoretic model for communicating sequential processes. The model underpins a process algebra that is very close to CSP. It differs from CSP "at the edges" for the purposes of creating an elegant algebra of communicating processes. The one significant difference is that we postulate additional distributive properties for external choice. The shape of the algebra that emerges suggests a lattice-theoretic model, in contrast to traditional trace-theoretic models. We show how to build the new model in a mathematically clean step-by-step process. The essence of our approach is to model simple processes (i.e. those without choice, parallelism, or recursion) as a poset S of sequences, and then order-embed S into a complete (and completely distributive) lattice called the *free completely distributive lattice* over S. We explain the technique in detail and show that the resulting model does indeed capture our algebra of communicating sequential processes. The focus of the paper is not on the algebra per se, but on the model and the soundness of the algebra.

Keywords: communicating sequential processes, denotational models, nondeterminacy.

1 Introduction

Process algebras are formally defined languages for the study of fundamental concepts in concurrent processes, including communication, synchronisation, nondeterminacy, abstraction, recursion, divergence, and deadlock. Among the best-known is CSP (*Communicating Sequential Processes*). Although CSP satisfies a large body of laws, the laws are not intended to be sufficient for everyday formal reasoning. Rather they "provide a useful way of gaining understanding and intuition about the intended meaning of constructs [and can] be useful in proofs about CSP processes" [6]. In fact, practitioners do not typically use algebraic means to reason about CSP code, but instead rely on model-checking approaches.

K. Barkaoui, A. Cavalcanti, and A. Cerone (Eds.): ICTAC 2006, LNCS 4281, pp. 123–137, 2006.

We embarked on constructing a practically useful algebra for communicating sequential processes that would be as CSP-like as we could make it, but which might depart from CSP "at the edges" if the algebra demanded it. We have constructed what seems to us to be a satisfactory algebra for a language that differs from CSP in one significant respect: we postulate that external choice enjoys the same distributive properties as internal choice. In contrast, external choice is somewhat less distributive in classical CSP. We speculate that this change does not impact significantly on the practice of writing CSP code, and that any drawbacks will be more than compensated for by the usefulness of the new algebra. However, the pros and cons of this are not the subject of the present paper. Our purpose here is to present a new model for an algebra of communicating sequential processes and show that the algebra is sound.

CSP has various models, all based on trace theory [6]. We have not adopted them because a more mathematically appealing approach suggested itself. In our algebra, internal and external choice are mathematical duals of one another, and so one would expect that they could be modelled by some lattice in which lattice meets and joins model internal and external choice, respectively. The creative part of our work lies in discovering this lattice and mapping all the terms and operators of communicating sequential processes into it. We think our model-building technique is sufficiently general to be useful in other contexts, in particular wherever nondeterministic choice is employed.

We construct the model, and show that the axioms of our algebra hold in it. Although it is not our purpose here to discuss our algebra per se, we will describe it to the extent that the reader needs to follow the soundness argument.

The model-building strategy can be broken into six steps, which we outline briefly:

Step 1. In the first step we model simple processes (or *proper* processes as we shall prefer to call them). These consist of processes that do not employ choice, parallelism, or recursion. Such processes are trivially modelled as sequences. The operations that apply to proper processes are easily modelled as operations on sequences. For example, sequential composition of processes is modelled more or less as sequence concatenation.

Step 2. Next we impose a partial ordering on the set of sequences. Our model building technique is indifferent as to how one chooses this ordering. One simply chooses it so that it captures the mathematical properties we want to hold, such as yielding the desired least fixpoint when modelling recursion, and ensuring that parallelism will have the properties we want.

Step 3. Next we introduce choice. We do so by finding an order-embedding of the sequence poset into some complete (and completely distributive) lattice such that lattice meets and joins model internal and external choice, respectively. The lattice we want turns out to be what is called the *free completely distributive (FCD) lattice* over the sequence poset.

Step 4. We have to "lift" the poset operators into the lattice. For example, sequence concatenation in the poset must be lifted into a similar operator in the lattice such that it preserves its behaviour. In addition, we have to give

the lifted operator additional behaviour to cater for the new lattice elements (these represent processes with choice). We develop a small suite of higher-order "lifting" devices that lift operators from a poset to the FCD lattice. For each operator on the sequence poset, we select an appropriate lifting into the lattice.
Step 5. In Step 5 we add parallelism. This has to be treated separately since unlike the other operators, parallelism can give rise to choice even when applied to proper processes.
Step 6. In the final step, we model recursively defined processes using completely standard fixed-point theory.

The rest of the paper is organised as follows:
Section 2. We describe the process algebra, and give its axioms.
Section 3. We give a poset model for proper processes.
Section 4. We show how to model choice using the FCD lattice construction.
Section 5. We give the model for the full language, and give the soundness result for the algebra.
Section 6. We give our conclusions and discuss related work.

2 The Process Algebra

Our language is based on that of CSP [6]. For brevity we will not consider all of the constructs of CSP. The subset we deal with is given in Table 1. Actually we will write $\sqcup$ rather than $\Box$ to emphasise the fact that our two choices are duals of one another. We will henceforth refer to our two choices as *demonic* and *angelic* choice instead of the traditional *internal* and *external* choice, respectively. Internal choice in CSP is precisely demonic choice as it occurs in other contexts (such as the refinement calculus [1]), and so the two names are interchangeable. The dual of demonic choice is typically called angelic choice in the literature, and so our terminology is in this regard consistent with established usage. However, we caution the reader against associating any "badness" with demonic choice, or any "goodness" with angelic choice. They are simply two choice operators that are mathematically dual. Henceforth, "external choice" will refer to CSP's version, and "angelic choice" will refer to ours. Angelic nondeterminacy is often associated with backtracking. However, our use of the term should not be taken to imply that an implementation is necessarily backtracking.

2.1 Proper Processes

We assume an alphabet of events denoted by Σ. There are two primitive processes: *SKIP* and *STOP*. *SKIP* denotes a process that has terminated successfully, and *STOP* denotes a process which has failed in some respect. We construct other simple processes from these base cases by *prefixing*, as in $a \rightarrow b \rightarrow c \rightarrow SKIP$ and $a \rightarrow b \rightarrow STOP$, where a, b and c are drawn from the alphabet of events.

For a set of events $A \subseteq \Sigma$, we define $\mathrm{Proc}(A)$ to be the set of processes constructed from *SKIP*, *STOP* and prefixing by events in A. We abbreviate $\mathrm{Proc}(\Sigma)$ by Proc. We call these simple processes *proper processes* and denote them by p, q, r.

Table 1. CSP

Σ	universal set of events
$STOP$	deadlocked process
$SKIP$	terminated process
$a \rightarrow P$	process P prefixed by event a
$\bigsqcap S$	internal choice of the terms in set S
$P \Box Q$	external choice of processes P and Q
$P \upharpoonright A$	restriction of P to the events in A
$P \,{}_A\|_B\, Q$	alphabetised parallel composition of P and Q with alphabets A and B
$\mu N.P$	recursively defined process

We partially order Proc by the *refinement order*, denoted by $\sqsubseteq$, and defined by the following axioms:

$$\begin{aligned}
&\text{(A1)}\ \ \sqsubseteq \text{ is a partial order}\\
&\text{(A2)}\ \ STOP \sqsubseteq p\\
&\text{(A3)}\ \ a \rightarrow p \sqsubseteq b \rightarrow q \ \Leftrightarrow\ (a = b) \wedge (p \sqsubseteq q)
\end{aligned}$$

where p, q denote proper processes and a, b denote (possibly equal) events. Actually, we need to assert that $p \sqsubseteq q$ does not hold unless it follows from the preceding axioms, and so we postulate in addition:

$$\text{(A4)}\ \ a \rightarrow p \not\sqsubseteq SKIP \not\sqsubseteq a \rightarrow p$$

(Note: We label the axioms of our algebra (A1), (A2), etc., and the theorems (other than axioms) by (T1), (T2), etc.)

It follows from (A1)–(A4) that *SKIP* and all proper processes ending in *SKIP* are maximal in the refinement order. A process which fails is refined by a process which can engage in the same events and then terminate, or one which can engage in the same events and then some further events. Otherwise processes are incomparable. Note that the refinement order is quite different from the common prefix order.

There are two operators which act on proper processes: restriction $p \upharpoonright A$ and sequential composition $p\,;q$. They are defined by the following axioms:

$$\begin{aligned}
&\text{(A5)}\ \ SKIP \upharpoonright A = SKIP\\
&\text{(A6)}\ \ STOP \upharpoonright A = STOP\\
&\text{(A7)}\ \ (a \rightarrow p) \upharpoonright A = \begin{cases} a \rightarrow (p \upharpoonright A) & \text{if } a \in A \\ p \upharpoonright A & \text{otherwise} \end{cases}\\
&\text{(A8)}\ \ SKIP\,; p = p\\
&\text{(A9)}\ \ STOP\,; p = STOP\\
&\text{(A10)}\ \ (a \rightarrow p)\,; q = a \rightarrow (p\,; q)
\end{aligned}$$

where A is a set of events, a is an event, and p, q are proper processes.

2.2 Choice Operators

For S a set of process terms, the term $\bigsqcap S$ denotes the demonic (or internal) choice of processes in S, and $\bigsqcup S$ denotes the angelic choice. We write $\sqcap$ and $\sqcup$ for the binary infix versions of $\bigsqcap$ and $\bigsqcup$, respectively. They are governed by the following axioms:

$$\text{(A11)}\ \ P \sqsubseteq Q \ \Leftrightarrow\ (\forall X \subseteq \text{Proc} \cdot \bigsqcap X \sqsubseteq P \Rightarrow \bigsqcap X \sqsubseteq Q)$$

$$\text{(A12)}\ \ P \sqsubseteq Q \ \Leftrightarrow\ (\forall X \subseteq \text{Proc} \cdot Q \sqsubseteq \bigsqcup X \Rightarrow P \sqsubseteq \bigsqcup X)$$

$$\text{(A13)}\ \ \bigsqcap S \sqsubseteq \bigsqcup X \ \Leftrightarrow\ (\exists P \in S \cdot P \sqsubseteq \bigsqcup X)$$

$$\text{(A14)}\ \ \bigsqcap X \sqsubseteq \bigsqcup S \ \Leftrightarrow\ (\exists P \in S \cdot \bigsqcap X \sqsubseteq P)$$

where P, Q are process terms, X is a set of proper processes, and S is a set of process terms. It is not easy to put an intuitive interpretation on these, and we suggest the reader does not try to do so. They have little role in the practical use of the algebra, but rather are used to establish a body of more practical theorems.

The preceding axioms also extend the refinement relation from proper to arbitrary processes, except that we need to postulate antisymmetry of refinement for arbitrary processes:

$$\text{(A15)}\ \ (P \sqsubseteq Q \wedge Q \sqsubseteq P) \ \Leftrightarrow\ P = Q$$

We can infer that if $R \subseteq S$ then $\bigsqcap S \sqsubseteq \bigsqcap R$, i.e. that refinement allows reduction in demonic choice (and dually an increase in angelic choice). We can also establish the classic lattice-theoretic relationship $P \sqsubseteq Q \Leftrightarrow P \sqcap Q = P \Leftrightarrow P \sqcup Q = Q$ where P and Q denote processes.

For empty choices, we define the abbreviations $\bot$ and $\top$ for $\bigsqcup \emptyset$ and $\bigsqcap \emptyset$, respectively. These satisfy $\bot \sqsubseteq P \sqsubseteq \top$ for all processes P; also $\sqcup$ and $\sqcap$ have units and zeros among $\bot$ and $\top$.

One of the most significant theorems of the algebra is that all processes can be expressed in a simple normal form. Before stating it, we introduce some additional notation: To express complex sets of process terms, we employ the set comprehension notation $\{x \in T \mid R \cdot P\}$, where R denotes a predicate and P denotes a term, in each of which x may occur free as a term of type T. This denotes the set of P's for each x in T that satisfy R . We write $\{x \in T \mid R\}$ as an abbreviation for $\{x \in T \mid R \cdot x\}$ and $\{x \in T \cdot P\}$ as an abbreviation for $\{x \in T \mid \text{true} \cdot P\}$.

Let us say that a term is *angelically proper* if it can be expressed as the angelic choice of a set of proper processes. It turns out that every process term can be expressed as the demonic choice of a set of angelically proper terms:

$$\text{(T1)}\ \ P \ =\ \bigsqcap \left\{ X \subseteq \text{Proc} \;\middle|\; P \sqsubseteq \bigsqcup X \cdot \bigsqcup X \right\}$$

for any process term P. (T1) says that any process P is equivalent to the demonic choice over all $\bigsqcup X$ where X ranges over those subsets of Proc satisfying $P \sqsubseteq$

$\bigsqcup X$. A term written this way is said to be in *demonic normal form* (there is also a dual *angelic normal form* which needn't concern us here).

2.3 Distribution Properties

We define prefixing and restriction to distribute over choice in their process arguments:

$$\text{(A16)}\ \ a \rightarrow (\bigsqcap S) \ = \ \bigsqcap \{P \in S \ \cdot \ a \rightarrow P\}$$
$$\text{(A17)}\ \ a \rightarrow (\bigsqcup S) \ = \ \bigsqcup \{P \in S \ \cdot \ a \rightarrow P\}$$
$$\text{(A18)}\ \ (\bigsqcap S) \upharpoonright A \ = \ \bigsqcap \{P \in S \ \cdot \ P \upharpoonright A\}$$
$$\text{(A19)}\ \ (\bigsqcup S) \upharpoonright A \ = \ \bigsqcup \{P \in S \ \cdot \ P \upharpoonright A\}$$

where a denotes an event, S a set of process terms and A a set of events. For sequential composition, which has two process arguments, we assert that it distributes over choice on the left:

$$\text{(A20)}\ \ (\bigsqcap S) ; P = \bigsqcap \{Q \in S \ \cdot \ Q ; P\}$$
$$\text{(A21)}\ \ (\bigsqcup S) ; P = \bigsqcup \{Q \in S \ \cdot \ Q ; P\}$$

where S is a set of process terms and P a process term. We also assert that it distributes over choice on the right, provided the left argument is proper:

$$\text{(A22)}\ \ p ; (\bigsqcap S) = \bigsqcap \{P \in S \ \cdot \ p ; P\}$$
$$\text{(A23)}\ \ p ; (\bigsqcup S) = \bigsqcup \{P \in S \ \cdot \ p ; P\}$$

where p is a proper process and S is a set of process terms. The requirement that the left argument in (A22) and (A23) be proper is a formal way of expressing that if both arguments of a sequential composition contain choice, distribution should happen on the left first and then on the right. Bearing in mind that the arguments can be expressed in normal form, (A20) to (A23) suffice to eliminate all choice from the arguments of a sequential composition, after which we can apply (A8) to (A10).

2.4 Alphabetised Parallel

There are several parallel operators in CSP, for example synchronising parallel, alphabetised parallel, interleaving, interface parallel, etc. We will describe the alphabetised parallel operator here, as it is typical. We give the operator three axioms, the first of which applies to angelically proper terms:

$$\text{(A24)}\ \ P \ {}_A\|_B \ Q \ = \ \bigsqcup \{p \in \text{Proc}(A \cup B) \mid p \upharpoonright A \sqsubseteq P \ \wedge \ p \upharpoonright B \sqsubseteq Q\}$$

where P, Q are angelically proper and A, B are sets of events. Its other two axioms assert that it distributes over demonic choice:

$$\text{(A25)}\ \ \bigsqcap S \ {}_A\|_B\ Q \ = \ \bigsqcap \left\{ P \in S \ \cdot \ P \ {}_A\|_B\ Q \right\}$$

$$\text{(A26)}\ \ P \ {}_A\|_B \bigsqcap S \ = \ \bigsqcap \left\{ Q \in S \ \cdot \ P \ {}_A\|_B\ Q \right\}$$

where S is a set of process terms, P, Q are arbitrary process terms and A, B are sets of events. The axioms are sufficient to prove that the operator is symmetric, associative (with appropriate alphabet adjustment) and satisfies a useful step law.

2.5 Recursively Defined Processes

Processes may be defined recursively, as in for example:

$$N \ = \ a \rightarrow N \ \sqcup \ SKIP$$

A recursive process definition is written $N = P$ where P is a process term which may contain free occurrences of N. The definition is *well-formed* only if P is monotonic in N and $P[STOP/N] \neq \bot$. ($P[Q/N]$ denotes the substitution of term Q for all free occurrences of N in P.) The monotonicity requirement excludes some unusual uses of the choice operators.

A well-formed recursive process definition $N = P$ defines a process $\mu N.P$ which satisfies the following axioms:

$$\text{(A27)}\ \ \mu N.P \ = \ P[(\mu N.P)/N]$$

$$\text{(A28)}\ \ P[Q/N] \sqsubseteq Q \ \Rightarrow \ (\mu N.P) \sqsubseteq Q$$

$$\text{(A29)}\ \ \mu N.P \neq \ \bot$$

where Q is a process such that $Q \neq \bot$.

An example of a recursively defined process is $\mu N.(a \rightarrow N)$. We can prove that it equals $\bigsqcup_{n \in \mathbb{N}} (a^n \rightarrow STOP)$ where a^n abbreviates a sequence of n as. Another example is $\mu N.(a \rightarrow N \sqcup SKIP)$ which can be shown to equal $\bigsqcup_{n \in \mathbb{N}} (a^n \rightarrow SKIP)$. We don't distinguish between deadlock and divergence in our model: the divergent process $\mu N.N$ equals $STOP$.

A recursive process definition may also involve an argument, as in for example:

$$\begin{aligned} \mathrm{COUNT} = \lambda n{:}\,\mathbb{N}.\ & \mathrm{up} \rightarrow \mathrm{COUNT}(n+1) \\ & \sqcup \textbf{if } n > 0 \textbf{ then } \mathrm{down} \rightarrow \mathrm{COUNT}(n-1) \textbf{ else } STOP \textbf{ fi} \end{aligned}$$

A parametrised recursive process definition is written $N = E$ where E is an abstracted process $(\lambda x{:}\,T.P)$ and T is some simple type (such as $\mathbb{N}$). The definition is *well-formed* only if N occurs in monotonic positions in P and for all $x \in T$, $E[(\lambda y : T \ \cdot \ STOP)/N](x) \neq \bot$. The defined process is written $(\mu N.E)$ and it satisfies the following axioms:

$$\text{(A30)}\ \ \mu N.E \ = \ E[(\mu N.E)/N]$$

$$\text{(A31)}\ \ (\forall\, x : T \cdot E[F/N](x) \sqsubseteq F(x)) \ \Rightarrow \ (\forall\, x : T \cdot (\mu N.E)(x) \sqsubseteq F(x))$$

$$\text{(A32)}\ \ \forall\, x : T \cdot (\mu N.E)(x) \ \neq \ \bot$$

where F is any function of type $T \rightarrow \mathrm{Proc}$ such that $(\forall\, x : T \cdot F(x) \neq \bot)$.

3 Modelling Proper Processes

Our first step in giving the semantics of our process algebra is to model the collection of proper processes as a poset.

Semantically, we won't distinguish between the type of events, Σ, and the set we use to model it. Similarly, we will allow an event a to model itself.

We model proper processes with a partially-ordered set denoted [Proc]. This consists of finite sequences of events which terminate in one of two ways: $\langle\rangle$ or Ω. Proper processes p have an interpretation in [Proc] which we denote $[p]$. They are interpreted as follows:

$$\begin{array}{lcl} [SKIP] & \triangleq & \langle\rangle \\ [STOP] & \triangleq & \Omega \\ [a \rightarrow p] & \triangleq & a{:}[p] \end{array}$$

where a is an event, p is a proper process and : is the cons operator on sequences. Let $\leq$ be the smallest partial order on [Proc] such that:

$$\begin{array}{l} (\forall\, u \in [\mathrm{Proc}] \cdot \Omega \ \leq \ u) \\ (\forall\, a \in \Sigma, u, v \in [\mathrm{Proc}] \cdot a{:}u \leq a{:}v \ \Leftrightarrow \ u \leq v) \end{array}$$

We use $\leq$ to model the refinement relation on proper processes.

The operators on proper processes are given meanings as monotonic operators on the poset [Proc]. Restriction is interpreted as:

$$[\restriction] \quad : \quad [\mathrm{Proc}] \times \mathbb{P}\Sigma \rightarrow [\mathrm{Proc}]$$

$$[\restriction](u, A) \ \triangleq \ \begin{cases} a{:}([\restriction](u', A)) & \text{if } u = a{:}u',\ a \in A \\ [\restriction](u', A) & \text{if } u = a{:}u',\ a \notin A \\ \langle\rangle & \text{if } u = \langle\rangle \\ \Omega & \text{if } u = \Omega \end{cases}$$

for all $u \in [\mathrm{Proc}]$ and $A \subseteq \Sigma$.

Sequential composition is interpreted as:

$$[;] \quad : \quad [\mathrm{Proc}] \times [\mathrm{Proc}] \rightarrow [\mathrm{Proc}]$$

$$[;](u, v) \ \triangleq \ \begin{cases} a{:}([;](u', v)) & \text{if } u = a{:}u' \\ v & \text{if } u = \langle\rangle \\ \Omega & \text{if } u = \Omega \end{cases}$$

for all $u, v \in [\mathrm{Proc}]$.

We can use $[\restriction]$ to give the interpretation of $\mathrm{Proc}(A)$:

$$[\mathrm{Proc}(A)] \ \triangleq \ \{x \in [\mathrm{Proc}] \mid [\restriction](x, A) \ = \ x\}$$

We have shown that the definitions we give for the poset and its operations are well-defined [8].

4 Modelling Unbounded Demonic and Angelic Choice

To model the choice operators, we will embed the poset in a complete lattice. There are many ways to embed a poset in a complete lattice, but the one we want is what's known as the *free completely distributive lattice* over a poset (FCD). The FCD lattice preserves the order of the original poset, is completely distributive, and has meets and joins that capture demonic and angelic choice, respectively.

4.1 Lattice Theory

Everything in this subsection is standard and is available in more detail in any standard text (such as [3,2]).

A *complete lattice* is a partially ordered set L such that every subset of L has a least upper bound and greatest lower bound. We will denote the order on the lattice by $\leq$. We denote the least upper bound of $S \subseteq L$ by $\bigvee S$ and the greatest lower bound by $\bigwedge S$. Least upper bounds are also called *joins* and greatest lower bounds are also called *meets*. A complete lattice is *completely distributive* iff joins distribute over meets and vice versa.

A function f from poset C to poset D is *monotonic* iff $x \leq_C y \Rightarrow f\,x \leq_D f\,y$ for all $x, y \in C$, and an *order-embedding* iff $x \leq_C y \Leftrightarrow f\,x \leq_D f\,y$ for all $x, y \in C$. An order-embedding from a poset C to a complete lattice is said to be a *completion* of C. We write $C \to D$ to denote the *space of monotonic functions* from C to D, which is a poset under the pointwise order. If L and M are complete lattices, then a function $f : L \to M$ is a *complete homomorphism* iff it preserves joins and meets, i.e. $f(\bigvee S) = \bigvee (fS)$ and $f(\bigwedge S) = \bigwedge (fS)$ for all $S \subseteq L$.

4.2 The Free Completely Distributive Lattice over a Poset

A completely distributive lattice L is called the *free completely distributive lattice over a poset* C iff there is a completion $\phi : C \to L$ such that for every completely distributive lattice M and function $f : C \to M$, there is a unique function $\phi^*_M f : L \to M$ which is a complete homomorphism and satisfies $\phi^*_M f \circ \phi = f$.

For any poset C, the free completely distributive lattice over C exists and is unique up to isomorphism [5]. It is written $\mathrm{FCD}(C)$. The completions $\phi : C \to \mathrm{FCD}(C)$ involved in the definition are not necessarily unique, but for each poset C, we assume that some such completion has been chosen.

We briefly offer some insight into the properties of FCD lattices. One of their most useful features is that each element of $\mathrm{FCD}(C)$ can be described as the meet of joins of subsets of $\phi\,C$, or the join of meets of subsets of $\phi\,C$. Another property of FCD lattices is that their bottom and/or their top element can be removed and the resulting structure is still a complete lattice. We will make use of this property when we model recursion.

Theorem 1. *Let ϕ be the FCD completion of C in* $\mathrm{FCD}(C)$*. Then, for all $x \in \mathrm{FCD}(C)$:*

$$x \;=\; \bigwedge \{X \subseteq \phi\,C \mid x \leq \bigvee X \;\cdot\; \bigvee X\} \;=\; \bigvee \{X \subseteq \phi\,C \mid \bigwedge X \leq x \;\cdot\; \bigwedge X\}$$

Proof. This is proved in [5].

Theorem 2. $\{x \in \mathrm{FCD}(C) \mid x \neq \bot\}$ *is a complete lattice under the inherited order from* $\mathrm{FCD}(C)$.

Proof. This is proved in [8].

4.3 Lifting Operators

Suppose that ϕ completes poset C in $\mathrm{FCD}(C)$. For each of the operators on C, we will want to lift them to corresponding operators on $\mathrm{FCD}(C)$. Since $\mathrm{FCD}(C)$ is a much richer space than C, it turns out that there are several options for how an operator is lifted.

To lift a unary operator $f : C \to C$, we define:

$$\begin{aligned} \mathcal{U} \quad &: (C \to C) \to (\mathrm{FCD}(C) \to \mathrm{FCD}(C)) \\ \mathcal{U}\, f &\triangleq \phi^*(\phi \circ f) \end{aligned}$$

As the following theorem shows, $\mathcal{U}f$'s behaviour on $\phi\, C$ corresponds to f's behaviour on C. Its behaviour outside $\phi\, C$ is determined by the fact that it distributes over meets and joins.

Theorem 3. *For all* $f : C \to C$, $x \in C$ *and* $X \subseteq \mathrm{FCD}(C)$*:*

$$\begin{aligned} (\mathcal{U}\, f)(\phi\, x) &= \phi\, (f\, x) \\ (\mathcal{U}\, f)(\bigwedge X) &= \bigwedge \{y \in X \;\cdot\; (\mathcal{U}\, f)\, y\} \\ (\mathcal{U}\, f)(\bigvee X) &= \bigvee \{y \in X \;\cdot\; (\mathcal{U}\, f)\, y\} \end{aligned}$$

We define the following two functions for when only one of the arguments of a binary operator is lifted to an FCD:

$$\begin{aligned} \mathcal{R} \quad &: (D \times C \to C) \to (D \times \mathrm{FCD}(C) \to \mathrm{FCD}(C)) \\ \mathcal{R}\, f &\triangleq \mathrm{uncurry}\, (\mathcal{U} \circ \mathrm{curry}\, f) \end{aligned}$$

$$\begin{aligned} \mathcal{L} \quad &: (C \times D \to C) \to (\mathrm{FCD}(C) \times D \to \mathrm{FCD}(C)) \\ \mathcal{L} \quad &\triangleq \mathrm{swap} \circ \mathcal{R} \circ \mathrm{swap} \end{aligned}$$

where curry, uncurry and swap are defined:

$$\begin{aligned} \mathrm{curry} \quad &= \lambda f : C \times D \to B \;\cdot\; \lambda x : C \;\cdot\; \lambda y : D \;\cdot\; f(x, y) \\ \mathrm{uncurry} &= \lambda f : C \to D \to B \;\cdot\; \lambda (x, y) : C \times D \;\cdot\; f\, x\, y \\ \mathrm{swap} \quad &= \lambda f : C \times D \to B \;\cdot\; \lambda (x, y) : D \times C \;\cdot\; f(y, x) \end{aligned}$$

Theorem 4. *For all* $f : D \times C \to C$, $g : C \times D \to C$, $x \in C$, $y \in D$ *and* $X \subseteq \mathrm{FCD}(C)$*:*

$$\begin{aligned} (\mathcal{R}\, f)\, (y\, ,\, \phi\, x) &= \phi\, (f(y\, ,\, x)) \\ (\mathcal{R}\, f)\, (y\, ,\, \bigwedge X) &= \bigwedge \{z \in X \;\cdot\; (\mathcal{R}\, f)\, (y\, ,\, z)\} \\ (\mathcal{R}\, f)\, (y\, ,\, \bigvee X) &= \bigvee \{z \in X \;\cdot\; (\mathcal{R}\, f)\, (y\, ,\, z)\} \end{aligned}$$

$$\begin{aligned} (\mathcal{L}\, g)\, (\phi\, x\, ,\, y) &= \phi\, (g(x\, ,\, y)) \\ (\mathcal{L}\, g)\, (\bigwedge X\, ,\, y) &= \bigwedge \{z \in X \;\cdot\; (\mathcal{L}\, g)\, (z\, ,\, y\})\} \\ (\mathcal{L}\, g)\, (\bigvee X\, ,\, y) &= \bigvee \{z \in X \;\cdot\; (\mathcal{L}\, g)\, (z\, ,\, y\})\} \end{aligned}$$

If both arguments of a binary operator are lifted to an FCD, one way of lifting the operator is called *left-first lifting*. Define the left-first lifting operator:

$$\begin{aligned}&\mathcal{B} \quad : \ (C \times C \to C) \to (\mathrm{FCD}(C) \times \mathrm{FCD}(C) \to \mathrm{FCD}(C))\\ &\mathcal{B}\, f \triangleq \mathrm{uncurry}(\phi^*(\mathcal{U} \circ (\mathrm{curry} f)))\end{aligned}$$

$\mathcal{B}\, f$ distributes over meets and joins on its left-hand argument first, and then on its right-hand argument.

Theorem 5. *For all $x, y \in C$, $X \subseteq \mathrm{FCD}(C)$ and $z \in \mathrm{FCD}(C)$:*

$$\begin{aligned}(\mathcal{B}\, f)\,(\phi\, x\,,\, \phi\, y) &= \phi\,(f(x\,,\, y))\\ (\mathcal{B}\, f)\,(\bigwedge X\,,\, z) &= \bigwedge \{w \in X \ \cdot \ (\mathcal{B}\, f)\,(w\,,\, z)\}\\ (\mathcal{B}\, f)\,(\bigvee X\,,\, z) &= \bigvee \{w \in X \ \cdot \ (\mathcal{B}\, f)\,(w\,,\, z)\}\\ (\mathcal{B}\, f)\,(\phi\, x\,,\, \bigwedge X) &= \bigwedge \{w \in X \ \cdot \ (\mathcal{B}\, f)\,(\phi\, x\,,\, w)\}\\ (\mathcal{B}\, f)\,(\phi\, x\,,\, \bigvee X) &= \bigvee \{w \in X \ \cdot \ (\mathcal{B}\, f)\,(\phi\, x\,,\, w)\}\end{aligned}$$

As well as left-first lifting, there are also right-first, meet-first and join-first lifting operators. These are not needed for the material considered here.

The theorems in this section are proved in [8].

5 The Model

The interpretation of the type Proc is given as:

$$[\![\mathrm{Proc}]\!] \ \triangleq \ \mathrm{FCD}([\mathrm{Proc}])$$

Let ϕ complete [Proc] in $[\![\mathrm{Proc}]\!]$. Each process term P has an interpretation $[\![P]\!] \in [\![\mathrm{Proc}]\!]$. Strictly, the interpretation is evaluated in an environment which assigns values to free variables. For the sake of readability, we will leave environments implicit whenever possible.

The constructors and operators of proper processes are interpreted as:

$$\begin{aligned}[\![SKIP]\!] &\triangleq \phi\,\langle\rangle\\ [\![STOP]\!] &\triangleq \phi\,\Omega\\ [\![a \to P]\!] &\triangleq [\![\to]\!](a, [\![P]\!])\\ [\![P \upharpoonright A]\!] &\triangleq [\![\upharpoonright]\!]([\![P]\!], A)\\ [\![P\,;Q]\!] &\triangleq [\![\,;]\!]([\![P]\!], [\![Q]\!])\end{aligned}$$

where a is an event, P, Q are process terms, A is a set of events and $[\![\to]\!]$, $[\![\upharpoonright]\!]$ and $[\![\,;]\!]$ are liftings of prefixing, restriction and sequential composition, respectively, defined as follows:

$$\begin{aligned}&[\![\to]\!] \ : \ \Sigma \times [\![\mathrm{Proc}]\!] \to [\![\mathrm{Proc}]\!]\\ &[\![\to]\!] \triangleq \mathcal{R}\,(:)\end{aligned}$$

$$\begin{aligned}&[\![\upharpoonright]\!] \ : \ [\![\mathrm{Proc}]\!] \times \mathbb{P}\Sigma \to [\![\mathrm{Proc}]\!]\\ &[\![\upharpoonright]\!] \triangleq \mathcal{L}\,[\upharpoonright]\end{aligned}$$

$$\begin{aligned}&[\![\,;]\!] \ : \ [\![\mathrm{Proc}]\!] \times [\![\mathrm{Proc}]\!] \to [\![\mathrm{Proc}]\!]\\ &[\![\,;]\!] \triangleq \mathcal{B}[;]\end{aligned}$$

where, in the definition of $[\![\rightarrow]\!]$, (:) is the cons operator : on [Proc] in prefix form.

The interpretation of a proper process lies in ϕ[Proc]. We interpret demonic and angelic choice as meet and join in the lattice, respectively.

5.1 Alphabetised Parallel Operator

To give the interpretation of the alphabetised parallel operator, we start with its behaviour on angelically proper processes. An angelically proper process can be represented by a set of elements of ϕ[Proc]. We define an operator $\langle\|\rangle$ which takes subsets of ϕ[Proc] as arguments. $\langle\|\rangle$ is defined as:

$$\langle\|\rangle \;:\; (\mathbb{P}(\phi[\text{Proc}]) \times \mathbb{P}\Sigma) \times (\mathbb{P}(\phi[\text{Proc}]) \times \mathbb{P}\Sigma) \rightarrow [\![\text{Proc}]\!]$$

$$\langle\|\rangle\,((X, A), (Y, B)) \triangleq \bigvee \{y \in \phi[\text{Proc}(A \cup B)] \mid [\![\Uparrow]\!](y, A) \leq \bigvee X \wedge [\![\Uparrow]\!](y, B) \leq \bigvee Y\}$$

We now give the denotation of $\|$ on $[\![\text{Proc}]\!]$:

$$[\![\,\|\,]\!] \;:\; ([\![\text{Proc}]\!] \times \mathbb{P}\Sigma) \times ([\![\text{Proc}]\!] \times \mathbb{P}\Sigma) \rightarrow [\![\text{Proc}]\!]$$

$$[\![\,\|\,]\!]\,((x, A), (y, B)) \triangleq \bigwedge\{X, Y \subseteq \phi[\text{Proc}] \mid x \leq \bigvee X \wedge y \leq \bigvee X \;\cdot\; \langle\|\rangle\,((X, A), (Y, B))\}$$

The interpretation of parallel compositions is as follows:

$$[\![P \;{}_A\|_B\; Q]\!] \triangleq [\![\|]\!]\,(([\![P]\!], A), ([\![Q]\!], B))$$

5.2 Recursive Process Definitions

Let $[\![\text{Proc}]\!]^-$ be $[\![\text{Proc}]\!]$ with $\perp$ removed. By Theorem 2, $[\![\text{Proc}]\!]^-$ is a complete lattice. Given a recursive process definition $N = P$, define f to be the following function:

$$f \;:\; [\![\text{Proc}]\!]^- \rightarrow [\![\text{Proc}]\!]^-$$
$$f \triangleq \lambda x : [\![\text{Proc}]\!]^- \;\cdot\; [\![P]\!]^x_N$$

where $[\![P]\!]^x_N$ denotes the interpretation of P when the environment associates x with N. For well-formed definitions, this is a well-defined monotonic function on the complete lattice $[\![\text{Proc}]\!]^-$. Therefore, by the standard Knaster-Tarski theory [7], generalised by Park [4], it has a least fixpoint μf. We define $[\![\mu N.P]\!] = \mu f$.

Given a parametrised recursive process definition $N = E$ where E has the form $(\lambda x{:}\, T.P)$, define f as follows:

$$f \;:\; ([\![T]\!] \rightarrow [\![\text{Proc}]\!]^-) \rightarrow ([\![T]\!] \rightarrow [\![\text{Proc}]\!]^-)$$
$$f \triangleq \lambda g : [\![T]\!] \rightarrow [\![\text{Proc}]\!]^- \;\cdot\; \lambda y : [\![T]\!] \;\cdot\; [\![P]\!]^{y\ g}_{x\ N}$$

For well-formed definitions, this is a well-defined monotonic function on the complete lattice $[\![T]\!] \rightarrow [\![\text{Proc}]\!]^-$. Therefore, it has a least fixpoint μf. We define $[\![\mu N.E]\!] \triangleq \mu f$.

5.3 Soundness

The soundness of the axioms is a corollary of the following two theorems. Each statement of the theorems justify a single axiom. The statements are marked with the corresponding axiom number.

Theorem 6. *For all* $z, w \in [\![\text{Proc}]\!]$, $X \subseteq \phi\,[\text{Proc}]$ *and* $S \subseteq [\![\text{Proc}]\!]$*:*

$$\text{(A11)}\ \ z \leq w \ \Leftrightarrow\ (\forall\, X \subseteq \phi\,[\text{Proc}] \cdot \bigwedge X \leq z \ \Rightarrow\ \bigwedge X \leq w)$$

$$\text{(A12)}\ \ z \leq w \ \Leftrightarrow\ (\forall\, X \subseteq \phi\,[\text{Proc}] \cdot w \leq \bigvee X \ \Rightarrow\ z \leq \bigvee X)$$

$$\text{(A13)}\ \ \bigwedge S \leq \bigvee X \ \Leftrightarrow\ (\exists\, z \in S \cdot z \leq \bigwedge X)$$

$$\text{(A14)}\ \ \bigwedge X \leq \bigvee S \ \Leftrightarrow\ (\exists\, z \in S \cdot \bigwedge X \leq z)$$

$$\text{(A15)}\ \ \leq \ \textit{is a partial order}$$

Proof. This follows from results in [5].

Theorem 7. *For all* $a, b \in \Sigma$, $x, y \in \phi\,[\text{Proc}]$, $A, B \subseteq \Sigma$, $S \subseteq [\![\text{Proc}]\!]$, $z \in [\![\text{Proc}]\!]$, $X, Y \subseteq \phi\,[\text{Proc}]$, $f \in [\![\text{Proc}]\!]^- \to [\![\text{Proc}]\!]^-$, $g \in ([\![T]\!] \to [\![\text{Proc}]\!]^-) \to ([\![T]\!] \to [\![\text{Proc}]\!]^-)$, *and* $h \in [\![T]\!] \to [\![\text{Proc}]\!]^-$*:*

$$\text{(A1)}\ \ \leq \ \textit{is a partial order}$$

$$\text{(A2)}\ \ \phi\,\Omega \ \leq\ x$$

$$\text{(A3)}\ \ [\![\to]\!](a,x) \leq [\![\to]\!](b,y) \ \Leftrightarrow\ (a = b) \wedge (x \leq y)$$

$$\text{(A4)}\ \ [\![\to]\!](a,x) \ \not\leq\ \phi\,\langle\rangle \ \not\leq\ [\![\to]\!](a,x)$$

$$\text{(A5)}\ \ [\![\restriction]\!](\phi\langle\rangle, A) \ =\ \phi\,\langle\rangle$$

$$\text{(A6)}\ \ [\![\restriction]\!](\phi\Omega, A) \ =\ \phi\,\Omega$$

$$\text{(A7)}\ \ [\![\restriction]\!]([\![\to]\!](a,x), A) \ =\ \begin{cases} [\![\to]\!](a, [\![\restriction]\!](x,A)) & \textit{if } a \in A \\ [\![\restriction]\!](x,A) & \textit{otherwise} \end{cases}$$

$$\text{(A8)}\ \ [\![\,;]\!](\phi\,\langle\rangle, x) \ =\ x$$

$$\text{(A9)}\ \ [\![\,;]\!](\phi\,\Omega, x) \ =\ \phi\,\Omega$$

$$\text{(A10)}\ \ [\![\,;]\!]([\![\to]\!](a,x), y) \ =\ [\![\to]\!](a, [\![\,;]\!](x,y))$$

$$\text{(A16)}\ \ [\![\to]\!](a, \bigwedge S) \ =\ \bigwedge \{w \in S \ \cdot\ [\![\to]\!](a,w)\}$$

$$\text{(A17)}\ \ [\![\to]\!](a, \bigvee S) \ =\ \bigvee \{w \in S \ \cdot\ [\![\to]\!](a,w)\}$$

$$\text{(A18)}\ \ [\![\restriction]\!](\bigwedge S, A) \ =\ \bigwedge \{w \in S \ \cdot\ [\![\restriction]\!](w,A)\}$$

$$\text{(A19)}\ \ [\![\restriction]\!](\bigvee S, A) \ =\ \bigvee \{w \in S \ \cdot\ [\![\restriction]\!](w,A)\}$$

$$\text{(A20)}\ \ [\![\ ;]\!](\bigwedge S, z) \ =\ \bigwedge \{w \in S \ \cdot\ [\![\ ;]\!](w,z)\}$$

$$\text{(A21)}\ \ [\![\ ;]\!](\bigvee S, z) \ =\ \bigvee \{w \in S \ \cdot\ [\![\ ;]\!](w,z)\}$$

$$\text{(A22)}\ \ [\![\ ;]\!](x, \bigwedge S) \ =\ \bigwedge \{w \in S \ \cdot\ [\![\ ;]\!](x,w)\}$$

$$\text{(A23)}\ \ [\![\ ;]\!](x, \bigvee S) \ = \ \bigvee \{w \in S \ \cdot \ [\![\ ;]\!](x, w)\}$$

$$\text{(A24)}\ \ [\![|\!|]\!]((\bigvee X, A), (\bigvee Y, B)) \ = \ \bigvee \{x \in \phi[\mathrm{Proc}(A \cup B)] \mid [\![\restriction]\!](x, A) \leq \bigvee X \ \wedge \ [\![\restriction]\!](x, B) \leq \bigvee Y\}$$

$$\text{(A25)}\ \ [\![|\!|]\!]((\bigwedge S, A), (z, B)) \ = \ \bigwedge \{w \in S \ \cdot \ [\![|\!|]\!]((w, A), (z, B))\}$$

$$\text{(A26)}\ \ [\![|\!|]\!]((z, A), (\bigwedge S, B)) \ = \ \bigwedge \{w \in S \ \cdot \ [\![|\!|]\!]((z, A), (w, B))\}$$

$$\text{(A27)}\ \ \mu f \ = \ f\,(\mu f)$$

$$\text{(A28)}\ \ z \neq \bot \ \wedge \ f\,z \leq z \ \Rightarrow \ \mu f \leq z$$

$$\text{(A29)}\ \ \mu f \ \neq \ \bot$$

$$\text{(A30)}\ \ \mu g \ = \ g\,(\mu g)$$

$$\text{(A31)}\ \ (\forall\, w \in [\![T]\!] \cdot g\,h\,w \ \leq \ h\,w) \ \Rightarrow \ (\forall\, w \in [\![T]\!] \cdot (\mu g)\,w \ \leq \ h\,w)$$

$$\text{(A32)}\ \ (\forall\, w \in [\![T]\!] \cdot (\mu g)\,w \ \neq \ \bot)$$

Proof. These results are proven in [8].

6 Conclusions

We have shown how to construct a model for an algebra of communicating sequential processes. The approach follows a mathematically clean step-by-step process which we speculate will apply whenever languages with choice are modelled. We first provide a poset which models the subset of the language without choice, parallelism or recursion. To model the whole language, we use the free completely distributive lattice over that poset. This is a suitable model for the choice operators, permits a very general model of the alphabetised parallel composition, and a natural definition of recursion. We have shown the algebra is sound by proving that the axioms hold in the model.

The algebra of communicating sequential processes we describe is very close to CSP: it differs mainly in the distributive properties of external choice. The model, however, is quite different from those of CSP and, in comparison, possesses some appealing qualities.

There exist several models for CSP [6], for example the failures-divergences model and the stable-failures model, all of which are based on trace theory. From a mathematical perspective, a complete, completely distributive lattice is a preferable structure to work with than the sets of pairs of traces employed by the standard CSP models, not least because it allows us to apply the tools of established lattice-theory.

Our prime motivation is the construction an algebra for communicating sequential processes which supports algebraic intuition and reasoning. The role a model plays in this context is to guarantee the soundness of the axioms. For this purpose, a single canonical model is desirable. This is what we have provided.

References

1. Ralph-Johan Back and Joakim von Wright. *Refinement Calculus — a systematic introduction*. Springer-Verlag, 1998.
2. Garrett Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society, 1995.
3. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.
4. P. Hitchcock and D. Park. Induction rules and termination proofs. In *IRIA Conference on Automata, Languages, and Programming Theory*, pages 225–251. North-Holland, Amsterdam, 1972.
5. Joseph M. Morris. Augmenting types with unbounded demonic and angelic nondeterminacy. In *Proceedings of the Seventh International Conference on Mathematics of Program Construction*, volume 3125, pages 274–288. Springer Verlag, 2004.
6. A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.
7. A. Tarski. A lattice-theoretical fixed point theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.
8. Malcolm Tyrrell, Joseph M. Morris, Andrew Butterfield, Arthur Hughes, and Wendy Verbruggen. A lattice-theoretic model for an algebra of communicating sequential processes: Definitions and proofs. Technical Report CA0206, School of Computing, Dublin City University, Dublin 9, Ireland, August 2006.