FINITE BUT UNBOUNDED DELAY IN SYNCHRONOUS CCS*

MICHAEL HUTH AND MARTA KWIATKOWSKA
School of Computer Science
University of Birmingham
Edgbaston, B15 2TT, United Kingdom
E-mail: mrh@cs.bham.ac.uk, mzk@cs.bham.ac.uk

Extended Abstract

ABSTRACT

We recast Milner's work on finite delay in synchronous CCS by giving a denotational semantics for admissibility of *infinite* computations on a bifinite domain $\mathcal{K}$. Using Abramsky's SFP domain $\mathcal{D}$ for bisimulation we obtain a fully abstract model $\mathcal{D} \times \mathcal{K}$ for an operational preorder which generalizes Milner's *fortification* and whose restriction to finite behavior corresponds to Abramsky's *finitary preorder*. The bifiniteness of $\mathcal{D} \times \mathcal{K}$ renders a program logic by considering its Stone dual.

1 Introduction

This paper mainly builds upon two pieces of work: Milner's synchronous CCS with finite delay [Mil82b] and Abramsky's domain equation for bisimulation [Abr91a]. The importance of synchronous CCS with finite delay stems from its ability to make fine-grained distinctions between *passive* resources, like an accessible register, and *active* ones, like agents that need to initiate some action in finite time. To clarify this point, consider an active agent δp which behaves like p, but can also delay by performing any number of silent actions 1. This can be defined by means of the equation

$$\delta p \equiv \mathsf{rec}\ x.1x + p \tag{1}$$

but, unfortunately, δp admits the infinite computation consisting entirely of delay steps (1^ω), and thus it cannot be guaranteed that p is *eventually* evoked. So, while δp models passive resources well, one still has to cater for the case of active agents. For that purpose, Milner introduced the operator ϵ, where ϵp

*Supported in part by EPSRC grant GR/K42028.

behaves like δp except that it *prevents* the infinite delay. Using finite delay, one can, for example, provide a more expressive coding of an asynchronous calculus within the synchronous one. The presence of finite but unbounded delay also allows for very natural solutions to certain fairness problems arising from the synchronous parallel operation. In fact, one may think of such a delay as a special *liveness property*.

Milner developed a theory of finite but unbounded delay with an accompanying notion of operational preorder (called *fortification*), an asymmetric and conservative extension of (strong) bisimulation in synchronous CCS. For guarded recursion he showed that recursive equations have least (but not always unique) fixed points with respect to this operational preorder, and that it is preserved by all the combinators of the calculus.

The notion of fortification crucially depends on that of an *admissible infinite computation* u for a process p. In [Mil82b] this had been formalized using prevention ordinals. However, Milner's paper does not give a denotational semantics of synchronous CCS with finite delay, and it is not at all clear how such a semantics should look like. For example, should it somehow be derived as a sort of limit from a denotational semantics for the finite computations as done in [Hen87, HK95][1] following an idea in [Win]? And if so, would we be able to "turn the handle" and extract a sound and complete program logic from a fully abstract denotational model? Unfortunately, the models in [Hen87, HK95] leave the realm of SFP domains for which there exists such a mechanical method for transforming a denotational semantics into an equivalent program logic [Abr91b].

In this extended abstract we sketch a fully abstract denotational model for an extension of partial fortification (fortification plus divergence) in a bifinite domain which then allows us to use the machinery of [Abr91b]. Since the category of bifinite domains is cartesian closed, this opens up the possibility of future work on finite delay in a higher-order setting. Our model is a product $\mathcal{D} \times \mathcal{K}$ where $\mathcal{D}$ is the initial solution for Abramsky's domain equation for bisimulation in [Abr91a]. We make use of the full abstraction result for the finitary preorder in [Abr91a]. But before we may re-use these established results, we have to deliver the following:

- A denotational semantics of admissible infinite computations with a bifinite domain $\mathcal{K}$ as target;
- a proof that this denotational semantics computes the same admissible infinite computations as the operational semantics based on prevention ordinals in [Mil82b];

[1] Such a derivation as a limit process has in fact *intrinsic* limitations [Mis95].

- the notion of a *generalized transition system* as an abstract concept for defining partial fortifications and an operational preorder which extends *partial bisimulations* to infinite computations; and
- a realization of $\mathcal{D} \times \mathcal{K}$ as such a generalized transition system together with a proof that it is internally fully abstract (i.e. the order on $\mathcal{D} \times \mathcal{K}$ coincides with our internal operational preorder).

The significance of this work lies foremost in giving a fully abstract denotational semantics for synchronous CCS with finite delay in a *bifinite*, and in particular, *algebraic* domain. This domain has a Stone dual from which one may read off the corresponding program logic. Another main contribution is the logical separation of finitely observable and truly infinite, "fair", behavior in the model resulting in a very elegant representation of the *environment* as $\mathcal{K}$, the infinite part of the model. By the same token, our model lets us "recycle" the full abstraction results of [Abr91a] for synchronous CCS on $\mathcal{D}$, the finitely observable part of the model.

Our results improve the work in [Hen83, Hen87], for we specify a *bifinite* model and we develop an asynchronous fair parallel from *finer* building blocks (synchronous product, sum and finite delay). Our work also goes beyond that in [HK95]; their model is *not* fully abstract and there proofs require substantial book-keeping of the synchronous product for synchronization algebras [Win84].

2 Synchronous CCS with finite delay

We study Milner's synchronous CCS with finite delay [Mil82b, Mil89] with the standard signature:

Definition 1 *Let $\Sigma = \{\Sigma_n\}_{n\in\omega}$ be the signature defined by*

$$\begin{array}{rcl} \Sigma_0 & \equiv & \{\mathsf{nil}\} \\ \Sigma_1 & \equiv & \{a\_ : a \in \mathsf{Act}\} \cup \{\_\lceil A : A \subseteq \mathsf{Act}\} \\ & & \cup \{\_[S] : S \textit{ is a monoid endomorphism on } \mathsf{Act}\} \\ & & \cup \{\epsilon\_\} \\ \Sigma_2 & \equiv & \{+, \times\} \\ \Sigma_n & \equiv & \emptyset, \qquad n > 2. \end{array}$$

where Act *is a non-trivial, countable, abelian group with identity* 1. *We extend* Σ *to* Σ' *by including variables* $x \in \Sigma'_0$ *ranging over a fixed and countable set* Var, *as well as fixed point operators: every* $x \in \mathsf{Var}$ *gives rise to an element* $\mathsf{rec}\ x$ *in* Σ'_1. *Further,* Σ'_1 *will also contain the* delay operator[2] δ.

[2]Strictly speaking, this operation is redundant as it could be defined by $\delta p \equiv \mathsf{rec}\ x.1x + p$ for all terms p.

Elements of the term algebra T_Σ are the finite terms of synchronous CCS with finite delay; elements of $T_{\Sigma'}$ are then the process terms of synchronous CCS with finite delay. The operational semantics is standard as in [Mil89]:

$$\frac{}{ap \to^a p} \ (\texttt{PREFIX})$$

$$\frac{p \to^a p', \ a \in A}{p\lceil A \to^a p'\lceil A} \ (\texttt{RESTRICTION})$$

$$\frac{p \to^a p'}{p[S] \to^{Sa} p'[S]} \ (\texttt{RELABELING})$$

$$\frac{p_1 \to^a p_1'}{p_1 + p_2 \to^a p_1'} \ (\texttt{SUM LEFT})$$

$$\frac{p_2 \to^a p_2'}{p_1 + p_2 \to^a p_2'} \ (\texttt{SUM RIGHT})$$

$$\frac{p_1 \to^a p_1', \ p_2 \to^b p_2'}{p_1 \times p_2 \to^{ab} p_1' \times p_2'} \ (\texttt{SYNCHRONOUS PRODUCT})$$

$$\frac{p[x \mapsto \mathsf{rec}\ x.p] \to^a p'}{\mathsf{rec}\ x.p \to^a p'} \ (\texttt{RECURSION})$$

with the two additional rules (`WAIT`) and (`FULFILL`) of [Mil82b]:

$$\frac{}{\epsilon p \to^1 \epsilon p} \ (\texttt{WAIT})$$

$$\frac{p \to^a p'}{\epsilon p \to^a p'} \ (\texttt{FULFILL})$$

The last two rules express the operational intuition that ϵp has the same dynamic capacity as the process p (`FULFILL`) but that it may choose to postpone its dynamics unboundedly but finitely often (`WAIT`). Using the operational rule (`RECURSION`), one readily notices that $\delta p \equiv \mathsf{rec}\ x.1x + p$ has the same *finite* behavior as ϵp; these processes are bisimilar. However, $\delta\, \mathsf{nil}$ is allowed to delay forever whereas $\epsilon\, \mathsf{nil}$ is not. So we need to re-define the notion of bisimulation in order to express such different capacities of performing *infinite* computations. Essentially, we need to rule out infinite computations

$$\epsilon p \to^1 \epsilon p \to^1 \epsilon p \to \dots \tag{2}$$

where each transition is justified by an instance of the (`WAIT`) rule. This will require a precise bookkeeping of such undesired *subcomputations* in a general context $\mathcal{C}(p_1, \dots, p_n)$ which is built from the only static combinators of this process calculus: synchronous product, relabeling and restriction. In [Mil82b] this is achieved via *prevention ordinals*; we extend his definition to include relabeling.

Definition 2 *[Mil82b] Let $\mathcal{P}$ denote the set of closed terms, i.e.,* agents, *in $T_{\Sigma'}$. We define a partial function*

$$\mathsf{o}\colon \mathcal{P} \times \mathsf{Act}^\omega \rightharpoonup \mathsf{Ord}$$

into the class of ordinals. We then say that p admits the infinite computation *$u \in \mathsf{Act}^\omega$ if and only if the ordinal $\mathsf{o}(p, u)$ is undefined. We define $\mathsf{o}(p, u)$ to be the least ordinal λ such that (p, u) is contained in the set M_λ. The sequence of subsets $M_\lambda \subseteq \mathcal{P} \times \mathsf{Act}^\omega$ is defined by transfinite induction, setting*

$$M_0 = \{(\mathsf{nil}, u)\colon u \in \mathsf{Act}^\omega\} \tag{3}$$

and for $\lambda > 0$

- *$(ap, bu) \in M_\lambda$ if $a \neq b$ or ($a = b$ & $(p, u) \in M_\mu$ for some $\mu < \lambda$);*
- *$(p\lceil A, u) \in M_\lambda$ if $u \notin A^\omega$ or $(p, u) \in M_\mu$ for some $\mu < \lambda$;*
- *$(p[S], u) \in M_\lambda$ if $u \neq S^\omega v$ for all $v \in \mathsf{Act}^\omega$ or for all $v \in \mathsf{Act}^\omega$ with $u = S^\omega v$ we have $(p, v) \in M_\mu$ for some $\mu < \lambda$;*[3]
- *$(\epsilon p, u) \in M_\lambda$ if for all $m \in \omega$ such that $u = 1^m v$ we have $(p, v) \in M_\mu$ for some $\mu < \lambda$;*
- *$(p_1 + p_2, u) \in M_\lambda$ if $(p_i, u) \in M_{\mu_i}$ for some $\mu_i < \lambda$, $(i = 1, 2)$;*
- *$(p_1 \times p_2, u) \in M_\lambda$ if for all $v, w \in \mathsf{Act}^\omega$ with $u = v \cdot w$ we have $(p_1, v) \in M_\mu$ or $(p_2, w) \in M_\mu$ for some $\mu < \lambda$;*
- *$(\mathsf{rec}\ x.p, u) \in M_\lambda$ if $(p[x \mapsto \mathsf{rec}\ x.p], u) \in M_\mu$ for some $\mu < \lambda$.*

Given an agent $p \in \mathcal{P}$ we associate with it its set of admissible infinite computations

$$\mathtt{k}(p) \equiv \{u \in \mathsf{Act}^\omega\colon\ \mathsf{o}(p, u) \text{ is undefined}\}. \tag{4}$$

For example, we have $\mathtt{k}(\delta\, \mathsf{rec}\, x.ax) = \{1^\omega\} \cup \{1^m a^\omega\colon m \in \omega\}$ and $\mathtt{k}(\epsilon\, \mathsf{rec}\, x.ax) = \{1^m a^\omega\colon m \in \omega\}$. Before we introduce operational preorders for agents we state the usual definition of a *divergence predicate* [Abr91a]:

$$\frac{x \in \mathsf{Var}}{\mathsf{rec}\ x.x \uparrow}\ (\Omega - \mathtt{OMEGA})$$

$$\frac{p \uparrow}{p\lceil A \uparrow}\ (\Omega - \mathtt{RESTRICTION})$$

[3]The map $S^\omega\colon \mathsf{Act}^\omega \to \mathsf{Act}^\omega$ is defined by $S^\omega u = S u_0 S u_1 \cdots$.

$$\frac{p\uparrow}{p[S]\uparrow}\ (\Omega - \texttt{RELABELING})$$

$$\frac{p_1\uparrow}{p_1+p_2\uparrow}\ (\Omega - \texttt{SUM LEFT})$$

$$\frac{p_2\uparrow}{p_1+p_2\uparrow}\ (\Omega - \texttt{SUM RIGHT})$$

$$\frac{p_1\uparrow\ \text{ or } p_2\uparrow}{p_1\times p_2\uparrow}\ (\Omega - \texttt{SYNCHRONOUS PRODUCT})$$

$$\frac{p[x\mapsto \mathsf{rec}\ x.x]\uparrow}{\mathsf{rec}\ x.p\uparrow}\ (\Omega - \texttt{RECURSION})$$

We write $p \to p'$ if $p \to^a p'$ for some $a \in \mathsf{Act}$. The triple $(\mathcal{P}, \to, \uparrow)$ is a transition system in the sense of [Abr91a]. Following Hennessy's and Stirling's ideas of generalized transition systems [HS85], we define such a notion which suitably represents admissible infinite computations.

Definition 3 *A* generalized transition system $\mathcal{G}$ over Act *is a quadruple* $(S, \to, k, \uparrow)$*, where* S *is a set of states,* $\to$ *associates to every* $a \in \mathsf{Act}$ *a binary relation* $\to^a\ \subseteq S \times S$ *and* $\uparrow \subseteq S$ *is a predicate of divergence. Finally,* $k\colon S \to \mathcal{K}$ *is a function, where* $\mathcal{K}$ *is the bifinite domain of subsets of* Act^ω *ordered under* reverse *inclusion.*[4]

Notice that $\mathcal{K}$ has uncountably many compact elements as the set Act^ω is uncountable. Further, generalized transition systems depend on Act and its derived parameter $\mathcal{K}$.

The intuition behind a generalized transition system is that it provides an operational semantics for abstract states in terms of transitions labeled by elements of Act and that $k(s)$ represents the set of admissible *infinite* computations beginning in state s.

We define natural co-inductive relations on generalized transition systems and we list their usual properties.

Definition 4 *Let* $\mathcal{G} = (S, \to, k, \uparrow)$ *be a generalized transition system over* Act*. A* partial fortification *on* $\mathcal{G}$ *is a binary relation* $R \subseteq S \times S$ *such that* xRy *implies*

- $k(x) \supseteq k(y)$
- $\forall a \in \mathsf{Act}\ x \to^a x'$ *implies the existence of some* y' *such that* $y \to^a y'$ *and* $x'Ry'$

[4] Of course, Act^ω denotes the set of infinite strings over Act.

- $x{\downarrow}$ *implies* $[y{\downarrow}$ *&* $(\forall a \in \mathsf{Act}\ y \to^a y'$ *implies the existence of some* x' *such that* $x \to^a x'$ *and* $x'Ry')]$.[5]

For general reasons there exists the greatest partial fortification on $\mathcal{G}$*, denoted by* $\prec_{\mathcal{G}}$ *or* $\prec$ *if* $\mathcal{G}$ *is clear from the context; moreover,* $\prec$ *is a preorder.*

A relation R *satisfying just the latter two clauses above is called a* partial bisimulation *[Abr91a]; the greatest partial bisimulation always exists and is a preorder denoted by* $\leq^{\mathsf{B}}$*; it is contained in Abramsky's* finitary preorder $\leq^{\mathsf{F}}$ *[Abr91a]. The operational preorder we consider is*

$$p \lhd q \text{ if and only if } p \leq^{\mathsf{F}} q \ \&\ k(p) \supseteq k(q), \tag{5}$$

the intersection of the finitary preorder with refinement with respect to admissibility. Also define $p \blacktriangleleft q$ *if and only if* $p \leq^{\mathsf{B}} q \ \&\ k(p) \supseteq k(q)$*.* [6]

We take $\lhd$ as the basis for this work since it encompasses $\prec$ and $\blacktriangleleft$ while still enjoying all the nice properties of $\prec$, such as being a congruence and having least fixed points. The possibility of considering several plausible operational preorders seems to be inherent in studying transition systems with infinite computations. For example, in [Hen84], we find two operational preorders for a simple language based on finite automata with three different delay operators. Hennessy is able to specify a complete axiomatization for one of them, but only a sound one for the other.[7] This is to be compared with our full abstraction results in subsection 4.3.

There are two special examples of partial fortifications. First, if $k(s) = \mathsf{Act}^\omega$ for all $s \in S$, this is simply the notion of 'partial bisimulation'. Second, if k is arbitrary and $\uparrow = \emptyset$ then we arrive at Milner's 'fortification relation' [Mil82b], which he defined for the concrete instance $\mathtt{k}$.

3 Admissibility Denotationally

We now make the lattice $\mathcal{K}$ into a monotone Σ-algebra. Using initial algebra and least fixed point semantics we get a denotational semantics $[\![\,]\!]^{\mathcal{K}}\colon \mathcal{P} \to \mathcal{K}$. We prove that $[\![p]\!]^{\mathcal{K}} = \mathtt{k}(p)$ for all $p \in \mathcal{P}$.

3.1 The Σ*-algebra* $\mathcal{K}$

Definition 5

$$\mathsf{nil}^{\mathcal{K}} \equiv \emptyset$$

[5] Convergence $\downarrow$ is defined as $S \backslash \uparrow$.

[6] One readily sees that $p \prec q$ implies $p \blacktriangleleft q$ but we do not know whether the converse is true as well. Clearly, $\blacktriangleleft$ is *properly* contained in $\lhd$.

[7] He does not define a denotational semantics.

$$
\begin{array}{rcl}
a^{\mathcal{K}}F & \equiv & \{au\colon u\in F\}, \qquad a\in \mathsf{Act}\\
F\lceil A^{\mathcal{K}} & \equiv & F\cap A^{\omega}, \qquad A\subseteq \mathsf{Act}\\
F[S]^{\mathcal{K}} & \equiv & \{S^{\omega}u\colon u\in F\},\\
 & & S \textit{ is a monoid endomorphism on } \mathsf{Act}\\
F+^{\mathcal{K}}G & \equiv & F\cup G\\
F\times^{\mathcal{K}}G & \equiv & \{u\in \mathsf{Act}^{\omega}\colon \exists v,w\in \mathsf{Act}^{\omega}\ u=v\cdot w,\\
 & & v\in F \textit{ and } w\in G\}\\
\epsilon^{\mathcal{K}}F & \equiv & \{1^{n}u\colon n\in\omega, u\in F\}\\
\delta^{\mathcal{K}}F & \equiv & \{1^{\omega}\}+^{\mathcal{K}}\epsilon^{\mathcal{K}}F.
\end{array}
$$

Lemma 1 *The functions* $\mathsf{nil}^{\mathcal{K}}$, $a^{\mathcal{K}}$, $\lceil A^{\mathcal{K}}$, $+^{\mathcal{K}}$, $\epsilon^{\mathcal{K}}$ *and* $\delta^{\mathcal{K}}$ *are continuous. The functions* $[S]^{\mathcal{K}}$ *and* $\times^{\mathcal{K}}$ *are monotone. If* S *is injective then* $[S]^{\mathcal{K}}$ *is continuous as well.*

The denotational semantics of admissible infinite computations is specified in the standard way:

Definition 6 *Let* $\mathsf{Env}_{\mathcal{K}}$ *be* $\mathcal{K}^{\mathsf{Var}}$.

$$
\begin{array}{rcl}
[\![x]\!]^{\mathcal{K}}\rho & = & \rho(x), \qquad x\in\mathsf{Var}\\
[\![\mathsf{op}(p_1,\ldots,p_k)]\!]^{\mathcal{K}}\rho & = & \mathsf{op}^{\mathcal{K}}([\![p_1]\!]^{\mathcal{K}}\rho,\ldots,[\![p_k]\!]^{\mathcal{K}}\rho),\\
 & & \mathsf{op}\in\Sigma_k\\
[\![\mathsf{rec}\ x.p]\!]^{\mathcal{K}}\rho & = & \mu F\in\mathcal{K}.[\![p]\!]^{\mathcal{K}}\rho[x\mapsto F].
\end{array}
$$

where $\mu F\in\mathcal{K}.[\![p]\!]^{\mathcal{K}}\rho[x\mapsto F]$ *denotes the least fixed point of the monotone function* $F\mapsto[\![p]\!]^{\mathcal{K}}\rho[x\mapsto F]\colon\mathcal{K}\to\mathcal{K}$.

The map $\times^{\mathcal{K}}$ is not continuous in general. We have an example of processes q and $p_0\lhd p_1\lhd p_2\lhd\ldots$ such that $\bigcap_{k\geq 0}([\![p_k]\!]^{\mathcal{K}}\times^{\mathcal{K}}[\![q]\!]^{\mathcal{K}})$ differs from $(\bigcap_{k\geq 0}[\![p_k]\!]^{\mathcal{K}})\times^{\mathcal{K}}[\![q]\!]^{\mathcal{K}}$. Therefore, the least fixed point will not, in general, be reached in ω steps. Since we want to show that $\mathtt{k}(p)=[\![p]\!]^{\mathcal{K}}$ for all agents $p\in\mathcal{P}$, we need to realize $[\![\,]\!]^{\mathcal{K}}$ as the *transfinite* limit of approximating semantics as done in [Par80, Hen83]:

Definition 7 *We define a family of functions* $[\![\,]\!]^{\mathcal{K}}_{\alpha}\colon T_{\Sigma'}\to\mathsf{Env}_{\mathcal{K}}\to\mathcal{K}$, $\alpha\in\mathsf{Ord}$, *by transfinite induction on* α*:*

$$
\begin{array}{rcl}
[\![p]\!]^{\mathcal{K}}_{0}\rho & \equiv & \mathsf{Act}^{\omega}\\
[\![x]\!]^{\mathcal{K}}_{\alpha+1}\rho & \equiv & \rho(x), \qquad x\in\mathsf{Var},\ \alpha\in\mathsf{Ord}\\
[\![\mathsf{op}(p_1,\ldots p_k)]\!]^{\mathcal{K}}_{\alpha+1}\rho & \equiv & \mathsf{op}^{\mathcal{K}}([\![p_1]\!]^{\mathcal{K}}_{\alpha}\rho,\ldots,[\![p_k]\!]^{\mathcal{K}}_{\alpha}\rho),\\
 & & \mathsf{op}\in\Sigma_k
\end{array}
$$

$$\begin{aligned}
[\![\mathsf{rec}\ x.p]\!]^{\mathcal{K}}_{\alpha+1}\rho &\equiv [\![p[x \mapsto \mathsf{rec}\ x.p]]\!]^{\mathcal{K}}_{\alpha}\rho \\
[\![p]\!]^{\mathcal{K}}_{\lambda}\rho &\equiv \bigcap_{\alpha<\lambda} [\![p]\!]^{\mathcal{K}}_{\alpha}\rho, \\
& \quad \lambda \textit{ limit ordinal.}
\end{aligned}$$

One easily shows that $\lambda < \mu$ implies $[\![\,]\!]^{\mathcal{K}}_{\mu} \leq [\![\,]\!]^{\mathcal{K}}_{\lambda}$. Since $\mathcal{K}$ is a dcpo we then know that there exists some ordinal λ_0 such that $[\![\,]\!]^{\mathcal{K}}_{\lambda} = [\![\,]\!]^{\mathcal{K}}_{\lambda_0}$ for all $\lambda \geq \lambda_0$. Without loss of generality, we assume that λ_0 is a limit ordinal.

Proposition 1 *[Hen83] We have $[\![\,]\!]^{\mathcal{K}} = [\![\,]\!]^{\mathcal{K}}_{\lambda_0}$ for the λ_0 above.*

Before we compare $[\![p]\!]^{\mathcal{K}}$ with $\mathtt{k}(p)$, let us gain some intuition about the meaning of $[\![\mathsf{rec}\ x.p]\!]^{\mathcal{K}}$.

Example 1

1. *Let $F \in \mathcal{K}$ and $f\colon \mathcal{K} \to \mathcal{K}$ given by $f(G) = 1^{\mathcal{K}}G +^{\mathcal{K}} F$. The function f is continuous and $\delta^{\mathcal{K}}F$ is the least fixed point of f; $\epsilon^{\mathcal{K}}F$ is the* only *other fixed point of f; it is strictly greater than $\delta^{\mathcal{K}}F$. If x is not free in p then*
$$[\![\mathsf{rec}\ x.1x + p]\!]^{\mathcal{K}}\rho = \delta^{\mathcal{K}}[\![p]\!]^{\mathcal{K}}\rho. \tag{6}$$

2. *Here is an example that shows how finite terms are being dealt with using the logical nature of $\emptyset$ for the Σ-algebra $\mathcal{K}$. We compute $[\![\mathsf{rec}\ x.a(\epsilon x + b\,\mathsf{nil})]\!]^{\mathcal{K}}$. If $p \equiv a(\epsilon x + b\,\mathsf{nil})$ we have $[\![\mathsf{rec}\ x.a(\epsilon x + b\,\mathsf{nil})]\!]^{\mathcal{K}} = \mu F \in \mathcal{K}.[\![p]\!]^{\mathcal{K}}[x \mapsto F] = \bigcap_{n\in\omega} f_p^n(\mathsf{Act}^{\omega})$ where $f_p(F) = a^{\mathcal{K}}(\epsilon^{\mathcal{K}}F +^{\mathcal{K}} b^{\mathcal{K}}\mathsf{nil}^{\mathcal{K}}) = a^{\mathcal{K}}(\epsilon^{\mathcal{K}}F +^{\mathcal{K}} b^{\mathcal{K}}\emptyset) = a^{\mathcal{K}}(\epsilon^{\mathcal{K}}F +^{\mathcal{K}} \emptyset) = a^{\mathcal{K}}\epsilon^{\mathcal{K}}F$. Therefore, $[\![\mathsf{rec}\ x.a(\epsilon x + b\,\mathsf{nil})]\!]^{\mathcal{K}} = \bigcap_{n\in\omega}(a^{\mathcal{K}}\epsilon^{\mathcal{K}})^n\mathsf{Act}^{\omega}$ and so $u \in [\![\mathsf{rec}\ x.a(\epsilon x + b\,\mathsf{nil})]\!]^{\mathcal{K}}$ if and only if there exists a family of natural numbers $(k_n)_{n\in\omega}$ such that $u = a1^{k_0}a1^{k_1}\cdots a1^{k_m}\cdots$ as one would expect.*

3. *Another example is* $\mathsf{rec}\ x.(a\,\mathsf{nil} + x)$ *and* $a\,\mathsf{nil}$. *Clearly, these terms are bisimilar. We also have* $\mathsf{rec}\ x.(a\,\mathsf{nil}+x) \lhd a\,\mathsf{nil}$, *but* $a\,\mathsf{nil} \not\lhd \mathsf{rec}\ x.(a\,\mathsf{nil}+x)$ *as $[\![a\,\mathsf{nil}]\!]^{\mathcal{K}} = \emptyset$ and $[\![\mathsf{rec}\ x.(a\,\mathsf{nil} + x)]\!]^{\mathcal{K}} = \mathsf{Act}^{\omega}$.*

3.2 Full Abstraction for $\mathcal{K}$

Theorem 1 *For all agents $p \in \mathcal{P}$ we have*
$$\mathtt{k}(p) = [\![p]\!]^{\mathcal{K}}. \tag{7}$$

4 Our Fully Abstract Model

4.1 Denotational Semantics

We make substantial use of Abramsky's paper [Abr91a] and briefly recall his powerdomain construction. For an SFP domain D let $P^o[D]$ be $P[D] \cup \{\emptyset\}$, where $P[D]$ is the standard Plotkin powerdomain of D, see e.g. [AJ95]. The domain $P^o[D]$ is ordered by $X \sqsubseteq Y$ if and only if $X = \{\bot\}$ or $X \sqsubseteq_{EM} Y$, where the latter is the Egli-Milner order denoting that

$$\forall x \in X\, \exists y \in Y\ x \leq y$$
$$\forall y \in Y\, \exists x \in X\ x \leq y.$$

Note that $\emptyset$ is above $\{\bot\}$ but incomparable with all the other elements in $P^0[D]$. Thus we can view $P^0[D]$ as a coalesced sum of the two-point domain and the Plotkin powerdomain of D. This view will readily give us a locally-continuous functor $P^0[\,]$ which, given a strict, continuous map $f\colon D \to E$, renders a continuous function $P^0[f]\colon P^0[D] \to P^0[E]$ with $P^0[f](\emptyset) = \emptyset$. The separated sum (or copower) $\Sigma_{a\in\mathsf{Act}}$ is locally-continuous and maps strict maps to strict maps. Using the standard machinery for solving recursive domain equations in SFP, Abramsky obtains an initial solution $\mathcal{D}$ for the following domain equation:

$$D \cong P^0[\Sigma_{a\in\mathsf{Act}} D]. \tag{8}$$

Following Abramsky's *principle of specification structures* [Abr95], we completely separate the finitely observable part (safety specification) from the infinite one (liveness) by considering

$$\mathcal{E} = \mathcal{D} \times \mathcal{K}. \tag{9}$$

We have already made $\mathcal{K}$ into a monotone Σ-algebra. Since ϵp and δp have the same operational semantics (=finitely observable behavior), we are compelled to define

$$\epsilon^{\mathcal{D}} \equiv \delta^{\mathcal{D}}\colon \mathcal{D} \to \mathcal{D} \tag{10}$$

where the latter is expressible as

$$\delta^{\mathcal{D}} d = \mu e \in \mathcal{D}.1^{\mathcal{D}} x +^{\mathcal{D}} d\, \rho[x \mapsto e]. \tag{11}$$

Thus we can think of $\mathcal{D}$ as a continuous Σ-algebra, which is essentially the one in [Abr91a] to which we just added a new function symbol ϵ modeled by the denotation of δ, i.e., via recursion. Therefore we can understand $\mathcal{E}$ as the *product* of the two Σ-algebras $\mathcal{D}$ and $\mathcal{K}$. In particular, we have a unique denotational semantics $[\![\,]\!]^{\mathcal{D}}\colon \mathcal{P} \to \mathcal{D}$ induced by the standard denotational semantics $[\![\,]\!]^{\mathcal{D}}\colon T_{\Sigma'} \to \mathsf{Env}_{\mathcal{D}} \to \mathcal{D}$.

To give some concrete examples, for $d, e \in \mathcal{D}$, $a \in \mathsf{Act}$ and $F, G \in \mathcal{K}$ we have

$$\begin{array}{rcl} \mathsf{nil}^{\mathcal{E}} & \equiv & (\emptyset, \emptyset) \\ (d, F) +^{\mathcal{E}} (e, G) & \equiv & (d \uplus e, F +^{\mathcal{K}} G) \\ a^{\mathcal{E}}(d, F) & \equiv & (\{\!|\langle a, d\rangle|\!\}, a^{\mathcal{K}} F). \end{array}$$

See [Abr91a] for the remaining, more complex, Σ-operations on $\mathcal{D}$.

Definition 8 *Let* $\mathsf{Env}_{\mathcal{D}}$ *be* $\mathcal{D}^{\mathsf{Var}}$ *and* $\mathsf{Env}_{\mathcal{E}} \equiv \mathcal{E}^{\mathsf{Var}}$.

$$\begin{array}{rcl} [\![x]\!]^{\mathcal{D}}\rho & = & \rho(x), \qquad x \in \mathsf{Var} \\ [\![\mathsf{op}(p_1,\ldots,p_k)]\!]^{\mathcal{D}}\rho & = & \mathsf{op}^{\mathcal{D}}([\![p_1]\!]^{\mathcal{D}}\rho,\ldots,[\![p_k]\!]^{\mathcal{D}}\rho), \\ & & \mathsf{op} \in \Sigma_k \\ [\![\mathsf{rec}\ x.p]\!]^{\mathcal{D}}\rho & = & \mu d \in \mathcal{D}.[\![p]\!]^{\mathcal{D}}\rho[x \mapsto d]. \end{array}$$

For all $p \in T_{\Sigma'}$ *and all* $\langle \rho_1, \rho_2 \rangle \in \mathsf{Env}_{\mathcal{E}}$ *we set*

$$[\![p]\!]^{\mathcal{E}}\langle \rho_1, \rho_2 \rangle = \langle [\![p]\!]^{\mathcal{D}}\rho_1, [\![p]\!]^{\mathcal{K}}\rho_2 \rangle. \tag{12}$$

4.2 $\mathcal{E}$ as a generalized transition system

We make $\mathcal{D}$ and $\mathcal{K}$ into generalized transition systems and view $\mathcal{E} = \mathcal{D} \times \mathcal{K}$ as a product of such systems. In [Abr91a] we have the transition system $(\mathcal{D}, \rightarrow, \uparrow)$ where $d \rightarrow^a d'$ if and only if $\langle a, d' \rangle \in d$, and $d\uparrow$ if and only if $\perp \in d$. We think of this as a generalized transition system by considering $(\mathcal{D}, \rightarrow, k, \uparrow)$ where $k(d) = \mathsf{Act}^\omega$ for all $d \in \mathcal{D}$. This reflects our stance that $\mathcal{D}$ solely models *finitely observable* behavior.

As for $\mathcal{K}$, we choose $(\mathcal{K}, \rightarrow, \mathrm{id}_{\mathcal{K}}, \emptyset)$ where we decree that for all $a \in \mathsf{Act}$ and all $F, G \in \mathcal{K}$ we have

$$F \rightarrow^a G \ \text{ if and only if } \ G = \{u \in \mathsf{Act}^\omega : au \in F\}. \tag{13}$$

This is a rigid consistency condition on the infinite and finite behavior of a process. The choice of id_K reflects that these sets 'denote themselves'. The choice of $\emptyset$ means that we can't specify divergence without having terms available. Also note that the transition relations on $\mathcal{K}$ nicely model the *environment*, for every state in $\mathcal{K}$ may respond with any action $a \in \mathsf{Act}$ in the most general way possible.[8] The reader is invited to check that the greatest partial fortification on $(\mathcal{K}, \rightarrow, \mathrm{id}_{\mathcal{K}}, \emptyset)$ equals $\supseteq$.

The map $a^{\mathcal{K}}: \mathcal{K} \rightarrow \mathcal{K}$ preserves all infima (=unions), so it has a left adjoint $(a^{\mathcal{K}})_*$ which preserves all suprema (co-limits). It is easy to see that the generalized transition system $(\mathcal{K}, \rightarrow, \mathrm{id}_{\mathcal{K}}, \emptyset)$ satisfies

$$F \rightarrow^a G \ \text{ if and only if } \ G = (a^{\mathcal{K}})_* F. \tag{14}$$

This gives a nice categorical explanation of the transitions on $\mathcal{K}$ as left adjoints of semantic prefixing.

[8] By abuse of notation, we will refer to this generalized transition system as $\mathcal{K}$.

Generalized transition systems are *structures* in the sense of model theory: $\rightarrow$ is a binary relation, $\uparrow$ is a predicate and each $u \in \mathsf{Act}^\omega$ corresponds to a predicate P_u, where $s \in P_u$ if and only if $u \in k(s)$. The category defined below is then the usual one for structures. Structures always have products.

Definition 9 *We define a category of generalized transition systems* $\mathsf{gts}(\mathsf{Act})$ *over* Act *which has generalized transition systems as objects and morphisms from* $(S_1, \rightarrow_1, k_1, \uparrow_1)$ *to* $(S_2, \rightarrow_2, k_2, \uparrow_2)$ *are total functions* $f\colon S_1 \rightarrow S_2$ *such that*

- $s \rightarrow^a s'$ *implies* $f(s) \rightarrow^a f(s')$
- $s{\downarrow_1}$ *implies* $f(s){\downarrow_2}$
- $u \in k_1(s)$ *implies* $u \in k_2(f(s))$.

Composition is component-wise and the identities are id_S.

It is readily seen that

$$(\{*\}, \{(*, *)\}_{a \in \mathsf{Act}}, k, \emptyset) \tag{15}$$

is a terminal object in $\mathsf{gts}(\mathsf{Act})$ where $k(*) = \mathsf{Act}^\omega$. More generally, if $(S_1, \rightarrow_1, k_1, \uparrow_1)$ and $(S_2, \rightarrow_2, k_2, \uparrow_2)$ are objects in $\mathsf{gts}(\mathsf{Act})$ we define $(S, \rightarrow, k, \uparrow)$ as

- $S = S_1 \times S_2$
- $(s_1, s_2) \rightarrow^a (s'_1, s'_2)$ if and only if $s_1 \rightarrow^a s'_1$ & $s_2 \rightarrow^a s'_2$
- $(s_1, s_2)\uparrow$ if and only if $s_1\uparrow$ or $s_2\uparrow$
- $k(s_1, s_2) = k_1(s_1) \cap k_2(s_2)$.

Only easily checks that the canonical projections are morphisms which make $(S, \rightarrow, k, \uparrow)$ into a product of $(S_1, \rightarrow_1, k_1, \uparrow_1)$ and $(S_2, \rightarrow_2, k_2, \uparrow_2)$ in $\mathsf{gts}(\mathsf{Act})$. Therefore, we have

$$(\mathcal{D}, \rightarrow, k, \uparrow) \times \mathcal{K} \cong (\mathcal{E}, \rightarrow, pr_2, \uparrow \circ \mathsf{pr}_1) \tag{16}$$

where pr_i, $(i = 1, 2)$, are the canonical projections such that $(d, F) \rightarrow^a (e, G)$ if and only if $d \rightarrow^a e$ in $\mathcal{D}$ and $F \rightarrow^a G$ in $\mathcal{K}$.

The product in (16) separates finitely observable from infinite behavior. This logical separation will be faithfully reflected in its greatest partial fortification.

Proposition 2 *Let* $(S, \rightarrow, k, \uparrow)$ *be a generalized transition system with* $k(s) = \mathsf{Act}^\omega$ *for all* $s \in S$ *and let* $\leq^{\mathsf{B}}$ *denote the greatest partial bisimulation on the transition system* $(S, \rightarrow, \uparrow)$. *The greatest partial fortification relation on the product* $(S, \rightarrow, k, \uparrow) \times \mathcal{K}$ *equals*

$$\{\langle p, q\rangle \in S \times S \colon p \leq^{\mathsf{B}} q \;\&\; k(p) \supseteq k(q)\}. \tag{17}$$

The proof of Proposition 2 crucially depends on the nature of our transition system on $\mathcal{K}$: all states $F \in \mathcal{K}$ may respond with every action $a \in \mathsf{Act}$ in a most liberal but deterministic way.

4.3 Full Abstraction

Using the full abstraction results for Abramsky's finitary preorder together with Theorem 1, we can prove full abstraction of our model with respect to $\lhd$.

Theorem 2 *The model $\mathcal{D} \times \mathcal{K}$ is internally fully abstract with respect to $\lhd$; it is further sound with respect to $\prec$ and fully abstract with respect to $\lhd$.*

5 Further Properties of our Model

5.1 A Fair Parallel

In [Mil82a] it has been shown that the composition operator '$|$' of the original asynchronous CCS in [Mil80] could be modeled in the synchronous calculus as $p \,|\, q \equiv p \times \delta q + \delta p \times q$. The discussion of *passive* versus *active* asynchronous agents in [Mil82b] suggests the short-comings of this definition and the need for introducing finite but unbounded delay syntactically (ϵ). Our model naturally supports a fair parallel $||$ which allows to distinguish between passive and active agents:

$$p \,||\, q \equiv p \times \epsilon q + \epsilon p \times q. \tag{18}$$

The equational properties of $||$ may be deduced from the properties of $+$ and ϵ in our fully abstract model. Since our semantics is a pairing, the equations have to be satisfied by $\mathcal{D}$ *and* by $\mathcal{K}$; the theory of $\mathcal{E}$ is the intersection of the theories of $\mathcal{D}$ and $\mathcal{K}$. In fact, $\mathcal{K}$ alone is a poor model (one for automata on infinite sequences), for $a(p+q) = ap + aq$ holds in it.

5.2 $\mathcal{D} \times \mathcal{K}$ in Logical Form

In [Abr91a] we find a complete account of the domain $\mathcal{D}$ in logical form. For a definition of $\times$ on domain pre-locales see [Abr91b]. We therefore restrict ourselves to expressing the domain $\mathcal{K}$ as a domain pre-locale A. Strictly speaking, we leave the category SFP in [Abr91b], for $\mathcal{K}$ is uncountably based, but as long as we allow our language to have uncountably many generators, the results in [Abr91b] will readily generalize to bifinite domains, which are *coherent algebraic spaces* [AJ95]. For illustration it suffices to specify the predicate C of [Abr91b]. We have

$$C_A = \{\{H \in \mathcal{K} \colon H \subseteq C\} \colon C \text{ is co-finite in } \mathcal{K}\}, \tag{19}$$

i.e., a typical element in C_A is of the form

$$\{H \in \mathcal{K} : F \cap H = \emptyset\}, \tag{20}$$

for some finite set $F \subseteq \mathsf{Act}^\omega$. This set corresponds to the property of not being allowed to do any infinite computations from a given finite set $F \subseteq \mathsf{Act}^\omega$.

Since our denotational semantics is actually only continuous in the first coordinate, we cannot extend the Stone duality on the level of objects to a duality on the level of morphisms in the usual way.

Acknowledgements

Samson Abramsky, Marcello Bonsangue, Matthew Hennessy, Alan Jeffrey, Achim Jung and Robin Milner have been extremely helpful and encouraging by providing detailed comments during various phases of this work.

References

[Abr91a] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.

[Abr91b] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51:1–77, 1991.

[Abr95] S. Abramsky. Semantics of interaction. Lecture Notes for the Summer School on Semantics and Logics of Computation, 25–29 September, Isaac Newton Institute,Cambridge University 1995.

[AJ95] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Clarendon Press, 1995.

[Hen83] M. C. B. Hennessy. Modelling Finite Delay Operators. Technical Report CSR-153-83, University of Edinburgh. Department of Computer Science, 1983.

[Hen84] M. C. B. Hennessy. Axiomatizing Finite Delay Operators. *Acta Informatica*, 21:61–88, 1984.

[Hen87] M. C. B. Hennessy. An Algebraic Theory Of Fair Asynchronous Communicating Processes. *Theoretical Computer Science*, 49:121–143, 1987.

[HK95] C. Hartonas and M. Kwiatkowska. Synchronization Trees and Fairness: A case study. In C. Hankin et al., editor, *Theory and Formal Methods 1994*. World Scientific Publishing, 1995. To appear.

[HS85] M. C. B. Hennessy and C. Stirling. The Power of the Future Perfect in Program Logics. *Information and Control*, 67(1–3):23–52, October/November/December 1985.

[Mil80] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[Mil82a] R. Milner. Calculi for synchrony and asynchrony. Technical Report CSR-104-82, University of Edinburgh, 1982.

[Mil82b] R. Milner. A finite delay operator in Synchronous CCS. Technical Report CSR-116-82, University of Edinburgh, 1982.

[Mil89] R. Milner. *Communication and Concurrency*. Series in Computer Science. Prentice-Hall International, 1989.

[Mis95] M. Mislove. Denotational Models for Unbounded Nondeterminism. In Maurice Nivat, editor, *Mathematical Foundations of Programming Semantics. Eleventh Annual Conference*, Electronic Notes in Theoretical Computer Science, pages 193–210, March 29–April 1 1995.

[Par80] D. M. Park. On the Semantics of Fair Parallelism. In D. Bjorner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 504–526. Springer Verlag, 1980.

[Win] G. Winskel. Generalized trees to give denotational semantics to finite delay. Unpublished Manuscript.

[Win84] G. Winskel. Synchronization trees. *Theoretical Computer Science*, 34:33–82, 1984.