# Categorical Models for Fairness and a Fully Abstract Presheaf Semantics of SCCS with Finite Delay*

Thomas T. Hildebrandt**

The IT University of Copenhagen, Denmark,
hilde@it-c.dk

**Abstract.** We present a presheaf model for the observation of *infinite* as well as finite computations. We apply it to give a *denotational* semantics of SCCS with finite delay, in which the meanings of recursion are given by *final* coalgebras and meanings of finite delay by *initial* algebras of the process equations for delay. This can be viewed as a first step in representing *fairness* in presheaf semantics. We give a concrete representation of the presheaf model as a category of *generalised synchronisation trees* and show that it is coreflective in a category of *generalised transition systems*, which are a special case of the general transition systems of Hennessy and Stirling. The open map bisimulation is shown to coincide with *extended bisimulation* of Hennessy and Stirling, which is essentially fair CTL$^*$-bisimulation. Finally we formulate Milners operational semantics of SCCS with finite delay in terms of generalised transition systems and prove that the presheaf semantics is *fully abstract* with respect to extended bisimulation.

## Introduction

When reasoning about and describing the behaviour of concurrent agents it is often the case that some infinite computations are considered *unfair* and consequently ruled out as being *inadmissible*. An economical way of studying this situation was proposed by Milner in [25] showing how to express a fair asynchronous parallel composition in his calculus SCCS (*Synchronous* CCS) by adding a *finite, but unbounded* delay operator. Syntactically, the finite delay of an agent $t$ is written $\epsilon t$. The intended semantics is that $\epsilon t$ can perform an unbounded number of 1-actions (delays) but *must eventually* perform an action, changing to an agent $t'$, if $t$ can perform this action and change to $t'$, or it *must eventually stop*, if $t$ cannot perform any actions. In other words, its actions are the same as for (the possibly infinite delay) $\delta t = \mathsf{rec}\, x.(1 : x + t)$, except that infinite unfolding of the recursion is not allowed.

To deal with agents in which only some infinite computations are admissible, one must readdress the issue of how to represent the behaviour of agents and so when two agents behave equally, i.e. they denote the same process. The approach used for CCS and SCCS, taking two agents to be equivalent if their derivation trees are strong bisimilar [23], will identify agents that only differ on whether some infinite computations are

* Full and revised version of a paper appearing in Proceedings of CTCS'99. Submitted to a journal.

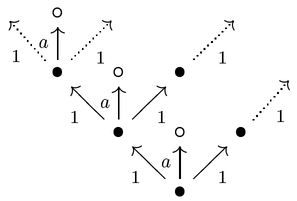
** Supported by the RCES project funded by the Danish Research Councils. This work was initiated during a stay at LFCS (University of Edinburgh, Scotland).

admissible or not, in particular $\epsilon t$ is identified with $\delta t$ for any term $t$. Moreover, both $\epsilon t$ and $\delta t$ should be solutions to the equation

$$x \cong (1 : x + t) \ , \tag{1}$$

so process equations will not anymore have *unique* solutions, as it is the case for CCS and SCCS (with guarded recursion).

In [25] Milner proposes a behavioral preorder called *fortification*, which is designed such that (1) it induces an equivalence which distinguishes the two notions of delay and coincides with strong bisimulation for "standard" agents, (2) recursive processes are *least* fixed points of the associated process equations and (3) the equivalence is a congruence with respect to all the operators of the language (assuming guarded recursion). This approach works reasonably, but is not completely satisfactory. As pointed out by Aczel in [1], the fortification equivalence makes some non desirable identifications of agents due to the fact that infinite computations are treated separately from finite computations. To illustrate this, one may consider the extension of strong bisimulation equivalence, obtained by requiring that for any two related states, the two sets of infinite action sequences labelling admissible infinite computations from each of the states must be identical. The resulting equivalence is included in fortification equivalence. Now, the two agents $\delta(a\colon 0 + \delta 0)$ and $\epsilon(a\colon 0 + \delta 0)$ (where 0 is the agent without any actions) have both a derivation tree with the shape




, (2)

where the underlying agents of black nodes are either the original agent or the agent $\delta 0$, for which $1^\omega$ is the *only* admissible infinite action sequence, and the underlying agents of white nodes are the agent 0, which has no action sequences at all. Consequently, the obvious isomorphism between the derivation trees of the two agents is a bisimulation satisfying the extra requirement given above, and thus the two agents are fortification equivalent. However, the first agent can delay infinitely *remaining* able to perform an a-action at any time, while the second agent must reach a state in which it cannot perform an a-action. The alternative proposed in [1] is a final-coalgebra semantics, giving rise to a bisimulation which indeed distinguishes the two agents given above. This bisimulation is closely related to the *extended* bisimulation introduced by Hennessy and Stirling in [14] for *general transition systems*, which is essentially *fair CTL*$^*$*-bisimulation* [10, 2], except being formulated for *edge* labelled structures.

The present paper is a revised version of chapter 7 in the authors PhD-thesis [15]. Its background is the work on presenting models for concurrency categorically as initiated by Winskel and Nielsen [30] and developed further in the work on bisimulation from open maps [19] and presheaf models for concurrency [4, 7, 16, 29]. Our goal is to extend the categorical approach (in which the issue of infinite computations and fairness has been absent so far) to models for infinite computations. As quality check, we want to apply the models to give both operational and denotational semantics for SCCS with finite delay, and capture a true branching equivalence that avoids the non-intuitive idenfications of fortification. As we will see, this goal can indeed be met.

One of the forces of describing models for concurrency within the language of category theory is that different models suitable for different purposes, can be formally related to each other. E.g. in [30] the category of synchronisation trees suitable for giving denotational semantics to CCS-like process calculi is shown to be a coreflective subcategory of the category of transition systems suited for operational semantics. Another force was added by the notion of *bisimulation from open maps* introduced in [19]. Here one gets an abstract behavioural equivalence by choosing a *path category*, or to be a bit more general [12], a functor from a category of *path shapes* to the model at issue, identifying the *observable computations* (in [19] assumed to be the inclusion of a subcategory). The open maps approach gained ground through the further development [4, 7, 6, 29, 4, 8] of the *presheaf* models for concurrency also proposed in [19]. Here one *starts* with a path category $\mathsf{P}$ (of non-empty path shapes) and then takes the category $\widehat{\mathsf{P}}$ of presheaves over $\mathsf{P}$ as model. The categorical justification [4, 8] for doing this, is the fact that $\mathcal{Y}^{\circ}_{\mathsf{P}_\perp} : \mathsf{P}_\perp \to \widehat{\mathsf{P}}$, the strict extension of the well-known Yoneda embedding, is the *free connected-colimit completion* of the category $\mathsf{P}_\perp$ obtained by adding a new intial object to $\mathsf{P}$. By the embedding $\mathcal{Y}^{\circ}_{\mathsf{P}_\perp}$ any presheaf model $\widehat{\mathsf{P}}$ comes with a *canonical* notion of bisimulation from open maps. In [7, 29, 4] it is shown that presheaf models themselves can be related within a category in which arrows are connected-colimit preserving functors, that such functors in fact *preserve* the canonical bisimulation and general techniques for their construction are provided.

Perhaps the simplest example of a presheaf model is the one equivalent to the category of ($Act$) labelled synchronisation trees, which is obtained from the path category of all finite, non-empty sequences of actions ordered by the usual prefix ordering. As shown in [19, 8, 7], the typical constructions of a CCS-like language can be expressed as functors preserving the canonical equivalence. As suggested in [19], it is natural to approach a generalisation of the categorical models to models for infinite computations, by studying presheaves, or sheaves, over the category of prefix ordered finite *and infinite* action sequences. In the present paper we show that with the help of a simple *Grothendieck topology* this gives indeed a suitable model for infinite computations, not as a category of sheaves, but as a category of *separated presheaves* [21]. A careful generalisation of the models of synchronisation trees and transition systems lifts the relationship between the “standard” finitary models to the infinitary models and gives a concrete representation of the presheaf model for infinite computations as generalised synchronisation trees, coreflective in a category of generalised transition systems. The generalised transition systems are defined as instances of the general transition systems of [14], and it turns out that the extended bisimulation defined in [14] coincides with the abstract bisimulation obtained from open maps. We end by showing how to express Milners [24] operational semantics of SCCS with finite delay in the generalised transition systems and give a denotational semantics in the presheaf model which we prove to be *fully abstract* with respect to *extended* bisimulation.

In all of the steps above we greatly benefit from the categorical presentation. *Unbounded non-determinism* is represented simply by (infinite) coproducts. By utilizing the general techniques from [4] we get very simple definitions of the denotations for all basic operators, for which congruence properties follow almost for free. As meanings of recursion we take *final coalgebras*, corresponding to *greatest* fixed points and the denotation of finite delay $\epsilon t$ is taken to be the initial algebra corresponding to the *least* fixed point of the process equation (1) given above. The categorical relationships between the different models and the general theory of bisimulation from open maps reduce the problem of relating the two semantics, proving full abstraction, to finding a specific open map within the category of generalised transition systems.

A number of papers [1, 18, 13, 14, 28] have already proposed denotational semantics for SCCS with finite delay, and in doing this, models for non-deterministic processes with infinite computations. As mentioned above, the approach we take is closely related to the work of Aczel in [1] and Hennessy and Stirling in [14]. The approach in [1] aims at a more general notion of fairness than finite delay. This appears to be at the cost that the admissible infinite computations are identified in a rather syntax dependent way, as opposed to our use of intitial algebras and final coalgebras. The semantics given in [18] is fully abstract with respect to the *fortification* equivalence, so it makes the non-intuitive identifications mentioned above. Moreover, it only covers *bounded non-determinism* as obtained from terms in which only a binary sum is allowed. The semantics of [13] is based on the fortification equivalence too. It is worth noting that for the models given in [18, 13, 28] the approximation order between elements is defined such that meanings of recursion can be given by *least* fixed points. This requires a *reverse* order between infinite computations, that is, the larger a process the fewer infinite computations it can make.

The structure of the paper is as follows. In Sec. 1 we give some preliminary definitions and recall the categorical concepts used in the paper. In Sec. 2 we recall the calculus SCCS [25], the finite delay operator and how to derive a fair parallel [24]. In Sec. 3 we introduce respectively the new presheaf model and the transition system models for infinite computations. Section 4 is devoted to the bisimulation obtained from open maps and its relationship to the extended bisimulation of [14]. In Sec. 5 we formulate Milner's operational semantics of SCCS with finite delay in terms of the generalised transition systems introduced in Sec. 3 and in Sec. 6 we give the presheaf semantics and the full abstraction result. Comments on future work is given in Sec. 7. The appendixes contain details on Grothendieck topologies and the proof of full abstraction.

## 1 Preliminaries

We assume a fixed set $Act$ of actions. Let $Act^+$ and $Act^\omega$ refer to the sets of respectively all finite and all infinite sequences of actions. We will let $\mathsf{Fin}$ and $\mathsf{Inf}$ refer to the two partial orders $(Act^+, \leq)$ and $(Act^+ \cup Act^\omega, \leq)$, where $\leq$ is the standard prefix order. These two partial orders will play the key role as path categories of presheaf models for the observation of respectively finite and possibly infinite computations. We will let roman letters range over elements (of some set) and greek letters range over sequences of elements (of some set). Let $|\alpha|$ denote the length of a sequence $\alpha$. For $\alpha, \alpha'$ such that $|\alpha| < \omega$ we write $\alpha\alpha'$ for the composition of the two sequences. If $j \in \omega$ and

$|\alpha| \geq j+1$, let $\alpha(j) = \alpha_0\alpha_1 \ldots \alpha_j$, i.e. the sequence of the first $j+1$ actions of $\alpha$. If $\beta \in S^+$ and $\beta \leq \alpha$ in $S^+ \cup S^\omega$, we will write $\beta \leq_f \alpha$ for $\beta$ is *finite and below* $\alpha$.

### 1.1 Categories of Transition Systems and Synchronisation Trees

We here repeat the definition of transition systems given in [30] and morphisms between such, which we will generalise to infinite computations in Sec. 3.2.

**Definition 1.** *A* transition system $T$ *with label set $Act$ is a quadruple* $(S_T, i_T, \to_T, Act)$*, where*

- $S_T$ *is a set of* states*,*
- $i_T \in S_T$ *is the* initial state *and*
- $\to_T \subseteq S_T \times Act \times S_T$ *is a* transition relation.

*As usual we write* $s \xrightarrow{a}_T s'$ *for* $\exists (s,a,s') \in \to_T$ *and let* $do\big((s,a,s')\big) = s$, $co\big((s,a,s')\big) = s'$, $act\big((s,a,s')\big) = a$. *We say a transition system is* reachable *if any state is reachable from the initial state. A* synchronisation tree *is a transition system for which the transition relation is acyclic and any state is reachable from the initial state by a* unique *sequence of transitions.*

*Transition systems (with label set $Act$) form the objects of a category* $\mathsf{TS}$*, with arrows being simulations. A simulation from $T$ to $T'$ is a mapping* $\sigma \colon S_T \to S_{T'}$ *of states, such that*

- $\sigma(i_T) = i_{T'}$ *and*
- $s \xrightarrow{a}_T s'$ *implies that* $\sigma(s) \xrightarrow{a}_{T'} \sigma(s')$*.*

*The synchronisation trees (with label set $Act$) induces a full subcategory* $\mathsf{ST}$ *of* $\mathsf{TS}$*.*

We define the set of finite or infinite computations of a transition system $T$ by

$$Comp(T) = \{\phi \in \to_T^* \cup \to_T^\omega \mid \forall 0 < j < |\phi|. co(\phi_{j-1}) = do(\phi_j)\},$$

and let $Comp_{fin}(T) = Comp(T) \cap \to_T^*$, i.e. the finite computations. We define the set of runs of a transition system $T$ by

$$Run(T) = \{\phi \in Comp(T) \mid do(\phi_0) = i_T\},$$

and let $Run_{fin}(T) = Run(T) \cap \to_T^*$.

As shown in [30], the category $\mathsf{ST}$ is a coreflective subcategory of the category $\mathsf{TS}$ of transition systems; the inclusion $\mathsf{ST} \hookrightarrow \mathsf{TS}$ has a right adjoint $unf \colon \mathsf{TS} \to \mathsf{ST}$ which acts on objects by *unfolding* the transition system. To be more precise, for a transition system $T$ with label set $Act$, its unfolding can be defined by

$$unf(T) = (Run_{fin}(T), \epsilon, \{\big(\phi, a, \phi(s,a,s')\big) \mid \phi, \phi(s,a,s') \in Run_{fin}(T)\}, Act),$$

where $\epsilon$ is the empty transition sequence. On morphisms, the unfolding takes a morphism between two transition systems $T$ and $T'$ with underlying map $\sigma \colon S_T \to S_{T'}$ to a morphism between $unf(T)$ and $unf(T')$ with underlying map $\sigma_* \colon Run_{fin}(T) \to Run_{fin}(T')$ defined by $\sigma_*(\phi) = \phi'$, such that $|\phi| = |\phi'|$ and for all $i < |\phi|$, if $\phi_i = (s,a,s')$ then $\phi'_i = \big(\sigma(s), a, \sigma(s')\big)$.

### 1.2 Bisimulation from Open Maps and Presheaf Models for Concurrency

The categorical presentation of models for concurrency comes with a general notion of *bisimulation from open maps* introduced in [19]. Given a model $\mathsf{M}$, the idea is to identify a functor $\mathcal{P}\colon \mathsf{P} \hookrightarrow \mathsf{M}$ from a category of *path shapes* to the model $\mathsf{M}$, identifying the *observable computations* (in [19] assumed to be the inclusion of a subcategory). A map $f\colon X \to Y$ in $\mathsf{M}$ is then defined to be *$\mathcal{P}$-open* ($\mathsf{P}$-open or just open if the embedding is clear from the context) if whenever for a morphism $m\colon P \to Q$ in $\mathsf{P}$ and morphisms $p$ and $q$ such that the diagram

$$\begin{tikzcd} \mathcal{P}P \arrow[r, "p"] \arrow[d, "\mathcal{P}m"'] & X \arrow[d, "f"] \\ \mathcal{P}Q \arrow[r, "q"] \arrow[ur, dotted, "h"] & Y \end{tikzcd}$$

commutes, there exists a morphism $h\colon \mathcal{P}Q \to X$ as indicated by the dotted line, making the two triangles commute. Two objects $X$ and $Y$ is then defind to be $\mathcal{P}$-bisimilar if they are related by a span of $\mathcal{P}$-open maps $f_1, f_2$, i.e.

$$\begin{tikzcd} & Z \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ X & & Y \end{tikzcd} .$$

It was shown in [19] that for $\mathcal{B}ran\colon \mathsf{Fin}_\perp \hookrightarrow \mathsf{TS}$ being the obvious embedding, mapping an action sequence to the corresponding single-branch transition system, $\mathcal{B}ran$-bisimulation coincides with the usual HM-bisimulation [25] on labelled transition systems. In subsequent work [9], a range of known bisimulations have been characterised as open map bisimulations. However, the freedom in how to choose the path-category seemed somehow unsatisfying.

In [19], presheaf categories were suggested as abstract models for concurrency, equipped with a *canonical* notion of bisimulation equivalence.

For $\mathsf{P}$ a small category, the category $\widehat{\mathsf{P}}$ of *presheaves over* $\mathsf{P}$ has as objects all functors $X\colon \mathsf{P}^{\mathsf{op}} \to \mathsf{Set}$ (where $\mathsf{Set}$ is the category of sets and functions) and as arrows natural transformations between such. Let us briefly repeat from [4, 8] the categorical justification for taking presheaf categories as models for concurrency. If we write $\mathsf{P}_\perp$ for the category obtained by adding a (new) initial object $\perp$ to $\mathsf{P}$, then $\widehat{\mathsf{P}}$ is the *free connected-colimit completion* of $\mathsf{P}_\perp$, i.e. the category obtained (up to equivalence) by freely adding all colimits of connected diagrams to $\mathsf{P}_\perp$. Any functor $F\colon \mathsf{P} \to \mathsf{Q}$ for $\mathsf{Q}$ a category having all connected colimits, can be extended freely (as a left Kan extension [20]) to a connected-colimit preserving functor $F_!\colon \widehat{\mathsf{P}} \to \mathsf{Q}$ making the diagram

$$\begin{tikzcd} \mathsf{P}_\perp \arrow[r, hook, "\mathcal{Y}^\circ_{\mathsf{P}_\perp}"] \arrow[dr, "F"'] & \widehat{\mathsf{P}} \arrow[d, "F_!"] \\ & \mathsf{Q} \end{tikzcd} \tag{3}$$

commute. The embedding $\mathcal{Y}^{\circ}_{\mathsf{P}_\perp} : \mathsf{P}_\perp \hookrightarrow \widehat{\mathsf{P}}$ is the strict extension of the well known Yoneda embedding $\mathcal{Y}_{\mathsf{P}} : \mathsf{P} \hookrightarrow \widehat{\mathsf{P}}$, taking $\perp$ to the empty presheaf. This embedding $\mathcal{Y}^{\circ}_{\mathsf{P}_\perp}$ provides the *canonical* choice of path shapes for a presheaf model $\widehat{\mathsf{P}}$, providing the canonical notion of bisimulation.[1]

The following proposition [5, 29, 4] is one of the most important results about open map bisimulation in presheaf models.

**Proposition 1.** *Let* $\mathcal{F} : \widehat{\mathsf{P}} \to \widehat{\mathsf{Q}}$ *be a connected-colimit preserving functor and* $m : X \to Y$ *a* $\mathcal{Y}^{\circ}_{\mathsf{P}_\perp}$*-open map in* $\widehat{\mathsf{P}}$. *Then* $\mathcal{F}m : \mathcal{F}X \to \mathcal{F}Y$ *is a* $\mathcal{Y}^{\circ}_{\mathsf{Q}_\perp}$*-open map in* $\widehat{\mathsf{Q}}$.

It follows that conected-colimit preserving functors between presheaf categories preserve the canonical notion of bisimulation.

As shown in [19], the category $\widehat{\mathsf{Fin}}$ is in fact equivalent to the category of synchronisation trees given above, and in [8, 7] it is shown that the typical constructions of a CCS-like language can be expressed as functors preserving the canonical equivalence. Below we repeat briefly the concrete representation of presheaves as transion systems as given in [31]. We will use it as a guide in Sec. 3 to help identify the right presheaf model for infinite computations.

**Notation 1** *If* $q \le p$ *in a partial order category* $\mathsf{P}$, *let* $[q, p]$ *denote the unique arrow in* $\mathsf{P}$ *and* $[p, q]$ *the unique arrow in* $\mathsf{P}^{\mathsf{op}}$. *We will employ the standard notation [21], writing* $x \cdot [q, p]$ *for the element* $X([p, q])x$, *i.e. the restriction of* $x$ *to the path* $q$.

For a presheaf $X$ in $\widehat{\mathsf{Fin}}$, its corresponding synchronisation tree $\mathcal{E}l(X) = (S_X, i_X, \to_X, Act)$ is constructed from *the category of elements* [21]. The set of states is given by the *elements* of $X$ with an initial state added as

$$S_X = \{(\alpha, x) \mid \alpha \in \mathsf{Fin} \text{ and } x \in X(\alpha)\} \cup \{i_X\},$$

and the transition relation is defined from the restriction action of the presheaf by

$$\to_X = \{((\alpha, x), a, (\alpha a, x')) \mid x' \cdot [\alpha, \alpha a] = x\} \cup \{(i_X, a, (a, x)) \mid x \in X(a)\}.$$

### 1.3 Initial Algebras and Final Coalgebras

Below we recall the categorical analogues of pre- and post-fixed points [3].

**Definition 2.** *Let* $F : \mathsf{P} \to \mathsf{P}$ *be an endofunctor on a category* $\mathsf{P}$. *A* co-algebra *for* $F$ *is a pair* $(p, m)$ *of an object and a morphism of* $\mathsf{P}$ *such that* $m : p \to F(p)$. *Dually, an* algebra *for* $F$ *is a pair* $(p, m)$ *such that* $m : F(p) \to p$. *The co-algebras of* $F$ *form the*

[1] In [19], focus was put on *rooted* presheaves over path categories with an initial object $\perp$, that is, presheaves for which $X(\perp)$ is the singleton set. The canonical bisimulation was taken to be that given by span of *surjective* $\mathcal{Y}$*-open maps*. It is easy to verify that the category of rooted presheaves over $\mathsf{P}_\perp$ is equivalent to the category $\widehat{\mathsf{P}}$. Moreover, surjective $\mathcal{Y}$-open maps between rooted presheaves in $\widehat{\mathsf{P}_\perp}$ corresponds via the equivalence eaxctly to $\mathcal{Y}^{\circ}$-open maps in $\widehat{\mathsf{P}}$.

*objects of a category* $\mathsf{F}_{\mathsf{coAlg}}$*, with arrows* $f\colon (p,m) \to (q,n)$ *being arrows* $f\colon p \to q$ *of* $\mathsf{P}$ *such that*

$$\begin{tikzcd} p \arrow[r, "m"] \arrow[d, "f"'] & F(p) \arrow[d, "F(f)"] \\ q \arrow[r, "n"] & F(q) \end{tikzcd}$$

*commutes. Dually, algebras for* $F$ *form the objects of a category* $\mathsf{F}_{\mathsf{Alg}}$.

Initial and final objects in $\mathsf{F}_{\mathsf{Alg}}$ and $\mathsf{F}_{\mathsf{coAlg}}$ are the categorical analogues of minimal and maximal fixed points of $F$.

**Lemma 1.** *Let* $F\colon \mathsf{P} \to \mathsf{P}$ *be an endofunctor on a category* $\mathsf{P}$*. If* $(p,m)$ *is an* initial algebra *for* $F$*, i.e. an initial object in the category of* $F$*-algebras, then* $m\colon F(p) \to p$ *is an isomorphism. Moreover if* $(q,n)$ *is another initial algebra for* $F$*,* $q$ *is isomorphic to* $p$*. The dual statement holds for* final co-algebras*. If* $F$ *has an initial algebra, let* $\mu F$ *denote the (unique upto isomorphism) initial algebra. Similarly, let* $\nu F$ *denote the final co-algebra of* $F$ *if it exists.*

The following lemma is a standard technique in proving existence of final co-algebras.

**Lemma 2.** *Let* $\mathsf{P}$ *be a category with terminal object* $\top$ *and* $F\colon \mathsf{P} \to \mathsf{P}$ *an endofunctor on* $\mathsf{P}$*. If the* $\omega^{op}$*-chain*

$$\top \leftarrow F(\top) \leftarrow F^2(\top) \leftarrow \ldots \leftarrow F^n(\top) \leftarrow \ldots$$

*has a limiting cone* $(P, \{p_n\colon P \to F^n(\top)\}_{n\in\omega})$ *and* $F$ *preserves this limit, i.e.*

$$(F(P), \{!\colon F(P) \to \top\} \cup \{F(p_n)\colon F(P) \to F^{n+1}(\top)\}_{n\in\omega})$$

*is a limiting cone too, then the unique mediating (iso)morphism* $m\colon P \to F(P)$ *is a final coalgebra.*

The above lemma is the dual of the following lemma for construction of initial algebras, as found in e.g. [3].

**Lemma 3.** *Let* $\mathsf{P}$ *be a category with initial object* $\bot$ *and* $F\colon \mathsf{P} \to \mathsf{P}$ *an endofunctor on* $\mathsf{P}$*. If the* $\omega$*-chain*

$$\bot \to F(\bot) \to F^2(\bot) \to \ldots \to F^n(\bot) \to \ldots$$

*has a colimit* $P$ *and* $F$ *preserves this colimit, then the unique mediating (iso)morphism* $m\colon F(P) \to P$ *is an initial algebra.*

Since limits are computed pointwise in a presheaf category $\widehat{\mathsf{P}}$, the terminal object in $\widehat{\mathsf{P}}$ is the presheaf $\top\colon \mathsf{P}^{op} \to \mathsf{Set}$ that yields the one element set (the terminal object in $\mathsf{Set}$) for any object $p$ in $\mathsf{P}$. Dually, the initial object $\bot\colon \mathsf{P}^{op} \to \mathsf{Set}$ is the empty presheaf, yielding the empty set for all objects $p$ in $\mathsf{P}$.

## 2 Synchronous CCS with Finite Delay

In this section we recall Milners calculus SCCS [25] of *synchronous* CCS and the definition of a fair parallel composition via a finite delay operator [24]. Assume a destinguished element $1 \in Act$ such that $(Act, \bullet, 1)$ is an Abelian monoid with 1 being the identity. The *basic* operators of SCCS are action prefixing, synchronous product, non-deterministic choice and restriction. Formally, the terms are given by

$$t ::= a\!:\!t \mid t_1 \times t_2 \mid \Sigma_{i \in I} t_i \mid t\lceil A,$$

where $a \in Act$, $A \subseteq Act$ and $I$ is an index set. With the basic operators we can build processes with only finite behaviour. As usual, we will write $0$ for an empty sum, omit the summation sign for a unary sum and write $t_1 + t_2$ for a binary sum.

To be able to define processes with possibly infinite runs, we add a recursion operator, extending the grammar by

$$t ::= \ldots \mid x \mid \mathsf{rec}\, x.t,$$

where $x$ is a process variable and $\mathsf{rec}\, x.$ binds the variable $x$ in $t$. We will let $\mathcal{T}$ refer to the set of closed terms of the calculus SCCS.

The rules given in Fig. 1 defines the operational semantics of SCCS, from which we get a *derivation transition system* for any closed term $t$ as defined below.

**Definition 3.** *Let $t$ be a term in $\mathcal{T}$. Then the* derivation transition system *for $t$ is the (reachable) transition system $D(t) = (S, t, \to_t, Act)$, where $S = \{t' \in \mathcal{T} \mid t \to^* t'\}$, i.e. all states reachable from $t$ by the relation $\to$ defined by the rules in Fig. 1 and $\to_t = \to \cap S \times Act \times S$.*

$$\frac{}{a\!:\!t \xrightarrow{a} t}, \quad \frac{t_j \xrightarrow{a} t'}{\Sigma_{i \in I} t_i \xrightarrow{a} t'} \quad (j \in I), \quad \frac{t_1 \xrightarrow{a} t_1' \quad t_2 \xrightarrow{b} t_2'}{t_1 \times t_2 \xrightarrow{a \bullet b} t_1' \times t_2'},$$

$$\frac{t \xrightarrow{a} t'}{t\lceil A \xrightarrow{a} t'\lceil A} \quad (a \in A), \quad \frac{t[\mathsf{rec}\, x.t/x] \xrightarrow{a} t'}{\mathsf{rec}\, x.t \xrightarrow{a} t'}.$$

**Fig.1.** Operational semantics of SCCS

Note that in the synchronous product, both processes must perform an action, and the resulting action is the monoid product of the two individual actions. Recursion acts by unfolding and $t[\mathsf{rec}\, x.t/x]$ is the usual substitution of $\mathsf{rec}\, x.t$ for the free variable $x$ in $t$.

An important derived operator introduced in [25] is the *delay* operator $\delta$. For a process $t$, define $\delta t = \mathsf{rec}\, x.(1\!:\!x + t)$. In the standard semantics, $\delta t$ is the (unique up to bisimulation) fixed point of the process equation

$$x \sim (1\!:\!x + t). \tag{4}$$

As an economical way to be able to express that some infinite runs are inadmissible, Milner introduces in [24] a *finite, but unbounded delay* operator $\epsilon$ (expectation). Its immediate actions are the same as for the derived delay operator, which can be described by the rules given in Fig. 2.

$$\frac{}{\epsilon t \xrightarrow{1} \epsilon t} \text{ (Wait)} \quad \text{and} \quad \frac{t \xrightarrow{a} t'}{\epsilon t \xrightarrow{a} t'} \text{ (Fulfill).}$$

**Fig.2.** Derivation Rules for Finite Delay

However, *infinite* waiting is ruled out as inadmissible. In other words, fulfillment of the delay is always expected. The idea is that finite delay is the *only* operator giving rise to inadmissible infinite runs. Recursion will as usual give rise to admissible infinite runs. This is sufficient to derive a *fair asynchronous* parallel composition. For processes $t$ and $t'$, the fair asynchronous parallel composition [24] of $t$ and $t'$ is defined by $t||t' = (\epsilon t \times t') + (t \times \epsilon t')$. The composition is asynchronous in the sense that one process can delay while the other progress; it is fair in the sense that no process can delay this way forever.

We will let SCCS$\epsilon$ and $\mathcal{T}_\epsilon$ refer to respectively the calculus SCCS extended with the finite delay operator $\epsilon$ and the set of terms of the extended calculus.

In the next section we will introduce two closely related categorical models, suitable for giving respectively denotational and operational semantics in which *inadmissibility* of infinite computations can be expressed.

## 3 Observing Infinite Computations

We approach a categorical model for infinite compuations by studying presheaves, and sheaves, over the path category $\mathsf{Inf}$ obtained by adding infinite paths to the path category $\mathsf{Fin}$. This fits with the spirit of [14], where experiments on systems are allowed to consist of *infinite* computations. Categorically, it can be seen as a completion of the path category with all directed colimits.

### 3.1 A Presheaf Model for Infinite Computations

To get a better understanding of presheaves $X \colon \mathsf{Inf}^{op} \to \mathsf{Set}$ in $\widehat{\mathsf{Inf}}$, one can try first to construct a synchronisation tree, as described in Sec. 1.2, for the finite part of $X$, i.e. the restriction of $X$ to $\mathsf{Fin}$. For $\alpha \in Act^\omega$, an element $x \in X(\alpha)$ will then specify a unique infinite path in the tree. To be more precise, if $\alpha \in Act^\omega$ and $x \in X(\alpha)$ then we will say that $x$ is a *limit point* of the infinite path given by the elements $x \cdot [\beta, \alpha]$ for $\beta \leq_f \alpha$, i.e. the restrictions of $x$ to finite observations. We wish to represent that an infinite path is *admissible* by the *presence* of such a limit point, and that it is *inadmissible* by the *absence* of a limit point. With this interpretation, the model is a bit too general; it allows an infinite path to have two or even more limit points, not representing anything more than if it had only one limit point. We take the subcategory of presheaves with atmost one limit point for any infinite sequence as our model. This category is not as ad hoc as it might seem. Actually, it comes about as the category of *separated presheaves* over $\mathsf{Inf}$ with respect to a simple Grothendieck topology for $\mathsf{Inf}$, which is often referred to as the sup topology. (In the standard terminology, the infinite paths and limit points are respectively *matching families* and (unique) *amalgations*).

**Definition 4.** *Let* $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ *denote the* separated presheaves*, which is the full subcategory of* $\widehat{\mathsf{Inf}}$ *induced by the presheaves* $X$ *satisfying that for all* $x, x' \in X(\alpha)$*,* $\alpha \in Act^\omega$

– *(Separated)*$(\forall \beta \leq_f \alpha . x \cdot [\beta, \alpha] = x' \cdot [\beta, \alpha]) \Rightarrow x = x'$.

Moreover, we can recover the category $\widehat{\mathsf{Fin}}$ (i.e. of synchronisation trees) within $\widehat{\mathsf{Inf}}$, as being equivalent to the category $\mathbf{Sh}(\widehat{\mathsf{Inf}})$ of *sheaves* over $\mathsf{Inf}$ for the same topology. In our case, a separated presheaf is a sheaf if it has *exactly* one limit point for any infinite path. Thus, a sheaf will correspond to a synchronisation tree in which *any* infinite path is admissible, i.e. a *limit closed* synchronisation tree. But this is just the standard interpretation made explicit.

**Proposition 2.** *The category* $\widehat{\mathsf{Fin}}$ *is equivalent to the category* $\mathbf{Sh}(\widehat{\mathsf{Inf}})$*, of sheaves over* $\mathsf{Inf}$ *with respect to the sup topology.*

Sheaves, separated presheaves and presheaves are known to be closely related and rich in structure [21, 32]. We will especially make use of the fact, that they are related by a sequence of reflections, i.e. the inclusions $\mathbf{Sh}(\widehat{\mathsf{Inf}}) \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}}) \hookrightarrow \widehat{\mathsf{Inf}}$ both have left adjoints (reflectors). In our case the reflections are particulary simple. The reflector $\mathsf{sp}\colon \widehat{\mathsf{Inf}} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ acts by unifying limit points that specify the same infinite path. The reflector from $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ to $\mathbf{Sh}(\widehat{\mathsf{Inf}})$ acts by completing with limit points of all infinte sequences.

We also have that the objects of $\mathsf{Inf}_\perp$ under the strict extension of the Yoneda embedding are sheaves.[2] Together with Prop. 2, this gives a formal relationship between the path category $\mathsf{Inf}_\perp$, the presheaf model $\widehat{\mathsf{Fin}}$ of finite observations and the models $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\widehat{\mathsf{Inf}}$ as summarized in the diagram below.

$$
\begin{tikzcd}
\widehat{\mathsf{Fin}} \arrow[d, leftrightarrow, "\cong"'] \arrow[dr, hook, "\mathsf{inf}"'] \arrow[dr, phantom, shift left=3, "\perp"] & & \\
\mathbf{Sh}(\widehat{\mathsf{Inf}}) \arrow[r, hook, shift right] \arrow[r, phantom, "\perp"] & \mathbf{Sp}(\widehat{\mathsf{Inf}}) \arrow[l, shift right] \arrow[r, hook, shift right] \arrow[r, phantom, "\perp"] & \widehat{\mathsf{Inf}} \arrow[l, shift right] \arrow[ull, "\mathsf{fin}"'] \\
\mathsf{Inf}_\perp \arrow[u, hook] \arrow[urr, hook, "\mathcal{Y}^\circ_{\mathsf{Inf}_\perp}"'] & &
\end{tikzcd}
\tag{5}
$$

[2] In fact the Grothendieck topology we use is the *canonical* topology for $\mathsf{Inf}$ [21], which simply means that it is the largest topology such that all the representables are sheaves.

Note that this also implies (a general fact) that the category $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ has all limits and colimits. In particular, it shows that limits are computed as in $\widehat{\mathsf{Inf}}$ and similarly for colimits, except for being followed by the reflector, identifying redundant limit points. As indicated in the diagram, we will let $\mathsf{fin} \dashv \mathsf{inf}$ refer to the reflection between $\widehat{\mathsf{Fin}}$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ obtained via the equivalecence between $\mathbf{Sh}(\widehat{\mathsf{Inf}})$ and $\widehat{\mathsf{Fin}}$.

For more details on Grothendieck topologies, sheaves and separated presheaves see [21]. The special, and simpler case for a Grothendieck topology on a partially ordered set is given in the appendix, together with the definition of the Grothendieck topology relevant for this paper.

### 3.2 Generalised Transition Systems and Synchronisation Trees

A generalised transition systems is a transition system in which the *admissible* infinite computations are represented explicitly. More precisely, we take a generalised transition system to be a transition system $T$ together with a set $C \subseteq Comp(T)$ such that $C = C^\bullet$, where $C^\bullet \subseteq Comp(T)$ be the least set including $C$ such that

C1: (composition) if $\phi, \phi' \in C^\bullet$ and $\phi\phi' \in Comp(T)$ then $\phi\phi' \in C^\bullet$,
C2: (pre- and suffix) if $\phi\phi' \in C^\bullet$ and $\phi$ is finite then $\phi, \phi' \in C^\bullet$ and
C3: (finite) $Comp_{fin}(T) \subseteq C^\bullet$.

The two first conditions ensure that the definition fits with that of *general transition systems* in [14]. The last condition restricts attention to the special case where any finite computation is admissible.

**Definition 5.** *A* generalised transition system (synchronisation tree) $G$ with label set *$Act$ is a five-tuple $(S_G, Adm_G, i_G, \rightarrow_G, Act)$, such that*

- *$T = (S_G, i_G, \rightarrow_G, Act)$ is a transition system (synchronisation tree), and*
- *$Adm_G \subseteq Comp(T)$ satisfies that $Adm_G = Adm_G{}^\bullet$.*

*We refer to $Adm_G$ as the set of* admissible computations *and say that G is* standard *if $Adm_G = Comp(T)$.*

For $G = (S_G, Adm_G, i_G, \rightarrow_G, Act)$ a generalised transition system let $fin(G) = (S_G, i_G, \rightarrow_G, Act)$ be the underlying transition system. Generalised transition systems (with label set $Act$) forms the objects of a category $\mathsf{GTS}$ with morphisms defined as follows.

**Definition 6.** *A morphism from a generalised transition system G to a generalised transition system $G'$ is given by a map $\sigma\colon S_G \rightarrow S_{G'}$ such that*

- *$\sigma(i_G) = i_{G'}$,*
- *$s \xrightarrow{a}_T s'$ implies that $\sigma(s) \xrightarrow{a}_{T'} \sigma(s')$ and*
- *$\sigma_\infty(Adm_G) \subseteq Adm_{G'}$,*

*where $\sigma_\infty$ is the extension of $\sigma$ to (possibly infinite) transition sequences defined by $\sigma_\infty(\phi) = \phi'$, such that $|\phi| = |\phi'|$ and for all $i < |\phi|$, if $\phi_i = (s, a, s')$ then $\phi'_i = \big(\sigma(s), a, \sigma(s')\big)$.*

As in the standard case, generalised synchronisation trees (with label set $Act$) induce a full subcategory $\mathsf{GST}$ of the category $\mathsf{GTS}$.

The following lemma gives some alternative definitions of morphisms between generalised transition systems. In particular it shows that the morphisms of $\mathsf{GTS}$ restrict to morphisms of the underlying transition systems, so the map $fin$ extends to a functor $fin\colon \mathsf{GTS} \to \mathsf{TS}$.

**Lemma 4.** *Let $\sigma\colon S_G \to S_{G'}$ be a map between the state sets of two generalised transition systems $G$ and $G'$. Then the following three conditions are equivalent*

1. *$\sigma\colon G \to G'$ is a morphism of generalised transition systems,*
2. – *$\sigma(i_G) = i_{G'}$ and*
   – *$\sigma_\infty(Adm_G) \subseteq Adm_{G'}$,*
3. – *$\sigma\colon fin(G) \to fin(G')$ is a morphism of transition systems and*
   – *$\sigma_\omega(Adm_G \backslash Comp_{fin}(G)) \subseteq Adm_{G'}$,*

In fact $fin\colon \mathsf{GTS} \to \mathsf{TS}$ is a reflector for the inclusion of $\mathsf{TS}$ into $\mathsf{GTS}$ that maps a plain transition system to the corresponding standard generalised transition system.

**Proposition 3.** *The functor $fin\colon \mathsf{GTS} \to \mathsf{TS}$ is a left adjoint to the inclusion $inf\colon \mathsf{TS} \hookrightarrow \mathsf{GTS}$ which maps a transition system $T = (S_T, i_T, \to_T, Act)$ to the standard generalised transition system $(S_T, i_T, \to_T, Comp(T), Act)$ and leaves morphisms unchanged.*

The coreflection between synchronisation trees and transition systems given in Sec. 1 generalises to one between between $\mathsf{GST}$ and a category $\mathsf{GTS}$.

**Proposition 4.** *The inclusion functor $\mathsf{GST} \hookrightarrow \mathsf{GTS}$ has a right adjoint $gunf\colon \mathsf{GTS} \to \mathsf{GST}$ such that the diagram*

$$\begin{tikzcd} \mathsf{GST} \arrow[d, "fin"'] & \mathsf{GTS} \arrow[l, "gunf"'] \arrow[d, "fin"'] \\ \mathsf{ST} & \mathsf{TS} \arrow[l, "unf"'] \end{tikzcd}$$

*commutes, where $unf$ is the unfolding of transition systems defined in 1.*

In fact we have that all four squares in the diagram

$$\begin{tikzcd} \mathsf{GST} \arrow[d, shift right=2, "fin"'] \arrow[r, hook, shift right=2] \arrow[r, phantom, "\top"] \arrow[d, phantom, "\dashv"] & \mathsf{GTS} \arrow[l, shift right=2, "gunf"'] \arrow[d, shift right=2, "fin"'] \arrow[d, phantom, "\dashv"] \\ \mathsf{ST} \arrow[u, hook, shift right=2] \arrow[r, hook, shift right=2] \arrow[r, phantom, "\top"] & \mathsf{TS} \arrow[l, shift right=2, "unf"'] \arrow[u, hook, shift right=2] \end{tikzcd}$$

commutes.

We will now generalise the equivalence between $\widehat{\mathsf{Fin}}$ and $\mathsf{ST}$, of which one direction was given by the category of elements construction described in Sec.1. This gives a concrete representation of presheaves in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ as generalised synchronisation trees.

To generalise the category of elements construction we use the following property stated in the lemma below: If every state in a generalised transition system is reachable, then the set of admissible computations is determined uniquely by the set of admissible *infinite runs*.

**Lemma 5.** *Let $T$ be a reachable transition system and $C \subseteq Comp(T)$. If $C = C^\bullet$ then there exists a unique set $A \subseteq Run(T) \backslash Run_{fin}(T)$ such that $C = A^\bullet$.*

The other direction is given by the embedding $\mathcal{GB}ran_+ \colon \mathsf{Inf} \hookrightarrow \mathsf{GST}$ which maps a non-empty, finite or infinite sequence to the corresponding single-branch standard generalised synchronisation tree. By implicitly using the inclusion of $\mathsf{GST}$ into $\mathsf{GTS}$ this gives a canonical functor [19] from $\mathsf{GTS}$ to $\widehat{\mathsf{Inf}}$, that maps a generalised transition system $G$ to the presheaf $\mathsf{GTS}[\mathcal{GB}ran_+(-), G]$. It is not difficult to check that this will always give a *separated* presheaf.

**Lemma 6.** *Let $G$ be a generalised transition system and $\mathcal{GB}ran_+ \colon \mathsf{Inf} \hookrightarrow \mathsf{GTS}$ the embedding described above. Then $\mathsf{GTS}[\mathcal{GB}ran_+(-), G]$ is a presheaf in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$.*

This functor gives us the other direction of the equivalence.

**Theorem 2.** *The categories $\mathsf{GST}$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ are equivalent. In one direction the equivalence is given by the canonical functor $\mathsf{sps} \colon \mathsf{GST} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ that maps a generalised synchronisation tree $G$ to the seperated presheaf $\mathsf{GST}[\mathcal{GB}ran_+(-), G]$. In the other direction, the equivalence is given by a functor $\mathcal{E}l \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathsf{GST}$ generalising the functor $\mathcal{E}l \colon \widehat{\mathsf{Fin}} \to \mathsf{ST}$ defined (on objects) in Sec. 1. For $X$ in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$, define $\mathcal{E}l(X) = (S, i, \to, Adm, Act)$, where $T = (S, i, \to, Act)$ is the synchronisation tree corresponding to the finite part of $X$ as defined in Sec. 1 and*

$$Adm = \{\phi \in Comp(T) \mid \exists \alpha \in Act^\omega \exists x \in X(\alpha). \forall j \in \omega. co(\phi_j) = \big(\alpha(j), x \cdot [\alpha(j), \alpha]\big)\}^\bullet.$$

Note that the functors $fin, inf$ restricted to synchronisation trees are just concrete representations of the reflection between $\widehat{\mathsf{Fin}}$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ given in Diagram (5).

## 4 Extended Bisimulation from Open Maps

As described in Sec. 1, we get a canonical notion of bisimulation from $\mathcal{Y}^\circ_{\mathsf{Inf}_\perp}$-open maps in the presheaf category $\widehat{\mathsf{Inf}}$. From Diagram (5) it follows that $\mathcal{Y}^\circ_{\mathsf{Inf}_\perp}$-bisimulation restricts to the subcategories $\mathbf{Sh}(\widehat{\mathsf{Inf}})$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ of sheaves and seperated presheaves. The category $\mathsf{Inf}_\perp$ embeds into the category of generalised transition systems by the functor $\mathcal{GB}ran \colon \mathsf{Inf}_\perp \hookrightarrow \mathsf{GST}$, which is simply the strict extension of $\mathcal{GB}ran_+ \colon \mathsf{Inf} \hookrightarrow \mathsf{GST}$ given above. This gives us a notion of $\mathsf{Inf}_\perp$-bisimulation for generalised transition systems, which we show coincides with *extended* bisimulation defined for general transition systems in [14].

It is worthwhile to remark, that extended bisimulation is essentially *fair CTL*-bisimulation* [11, 10, 2], except for being formulated for *edge* labelled structures. This means in particular, that extended bisimulation is decidable for finite generalised transition systems for which the set of infinite paths are given by e.g a Büchi- or Muller-condition [2].

We first give a characterisation of the $\mathsf{Inf}_\perp$-open maps of GTS that generalises the "zig-zag" morphisms in [19].

**Proposition 5.** *Let $T = (S_T, i_T, \to_T, Adm_T, Act)$ and $U = (S_U, i_U, \to_U, Adm_U, Act)$ be generalised transition systems and $\sigma : T \to U$. Then $\sigma$ is $\mathsf{Fin}_\perp$-open if and only if for all reachable states $s \in S_T$*

- *if $\sigma(s) \xrightarrow{a}_U s'_1$ then $s \xrightarrow{a}_T s_1$ and $\sigma(s_1) = s'_1$ for some state $s_1 \in S_T$,*

*and $\sigma$ is $\mathsf{Inf}_\perp$-open if and only if moreover*

- *if $\phi' \in Adm_U$ and $\phi' = \sigma(s) \xrightarrow{a_1}_U s'_1 \xrightarrow{a_2}_U s'_2 \xrightarrow{a_3}_U \ldots \xrightarrow{a_n}_U s'_n \xrightarrow{a_{n+1}}_U \ldots$ then there exists $\phi \in Adm$ such that $\phi = s \xrightarrow{a_1}_T s_1 \xrightarrow{a_2}_T s_2 \xrightarrow{a_3}_T \ldots \xrightarrow{a_n}_T s_n \xrightarrow{a_{n+1}}_T \ldots$ and for all $j \in \omega$, $\sigma(s_j) = s'_j$*

Now we give the definition of extended bisimulation from [14] reformulated as a relation between two generalised transition systems (exploting condition C3).

**Definition 7.** *([14]) Let $T$ and $T'$ be generalised transition systems. Then $T$ and $T'$ are* extended bisimilar *if there exists a relation $R \subseteq S_T \times S_{T'}$ such that $(i_T, i_{T'}) \in R$ and if $(s, s') \in R$ then*

E1. *if there exists a computation $\phi \in Adm_T$ s.t. $\phi_0 = s$, then there exists a computation $\phi' \in Adm_{T'}$ s.t. $|\phi| = |\phi'|$ and $\phi'_0 = s'$ and for $0 \le j < |\phi|$, $act(\phi_j) = act(\phi'_j)$ and $(\phi_j, \phi'_j) \in R$,*

E2. *if there exists a computation $\phi' \in Adm_{T'}$ s.t. $\phi'_0 = s'$, then there exists a computation $\phi \in Adm_T$ s.t. $|\phi| = |\phi'|$ and $\phi_0 = s$ and for $0 \le j < |\phi|$, $act(\phi_j) = act(\phi'_j)$ and $(\phi_j, \phi'_j) \in R$,*

Note that (by condition C3) extended bisimulation specialises to the standard HM-bisimulation on transition systems if E1 and E2 are restricted to sequences of length one. Also note that (by the conditions C1 and C2) one could equivalently have formulated the bisimulation considering only sequences being infinite or of length one. From these considerations and Prop. 5 it follows that extended bisimulation coincides with $\mathsf{Inf}_\perp$-bisimulation for generalised transition systems.

**Proposition 6.** *Let $G$ and $G'$ be generalised transition systems. Then $G$ and $G'$ are $\mathsf{Inf}_\perp$-bisimilar if and only if $G$ and $G'$ are extended bisimilar.*

It is an easy fact that $\mathsf{Inf}_\perp$-bisimulation in GST under the equivalence coincides with the canonical bisimulation in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$, so we get the following corollary.

**Corollary 1.** *Let $X$ and $X'$ be presheaves in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$. Then $X$ and $X'$ are $\mathcal{Y}^{\circ}_{\mathsf{Inf}_\perp}$-bisimilar if and only if $\mathcal{E}l(X)$ and $\mathcal{E}l(X')$ are extended bisimilar.*

From the coreflection given in the previous section and Lem. 6 in [19] it follows that two generalised transition systems are $\mathsf{Inf}_\perp$-bisimilar if and only if their unfoldings as generalised synchronisation trees are $\mathsf{Inf}_\perp$-bisimilar.

## 5 Operational Semantics

In this section we will express Milner's operational semantics of SCCS with finite delay [24] in terms of generalised transition system. First the two rules in Fig.2 are added to the rules of Fig.1. Next the inadmissible infinite computations are identified via the notions of waiting computations, subagents and subcomputations. Put briefly: A computation $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots$ of an agent $t_0$ is waiting if $t_i = \epsilon t$ for all $i$ and every transition is inferred *solely* from the (Wait) rule for finite delay. Agents $a : t$, $\mathsf{rec}\, x.t$, $\Sigma_{i \in I} t_i$ and $\epsilon t$ have only themselves as subagent, $t \lceil A$ has the subagents of $t$ and $t_1 \times t_2$ has the subagents of $t_1$ and $t_2$. Any computation of an agent $t$ is then inferred from computations of the subagents, which are referred to as subcomputations. A computation is defined to be admissible if it is finite or has no sequel (i.e. suffix) with an infinite waiting subcompuation.

To define a derivation transition system in which we can distinguish admissible from inadmissible infinite runs we thus need to record if the (Wait) rule was used to infer an action of a subagent. Consequently, we will annotate terms of the form $\epsilon t$ with a number $n \in \omega$ written $\epsilon_n t$, which indicates for how long they have been delaying. In the following $\mathcal{T}_\epsilon$ will generally refer to the set of annotated closed terms of SCCS$\epsilon$. Note that any function with domain $\mathcal{T}$ can be regarded as a function with domain $\mathcal{T}_\epsilon$ by discarding the annotations. For simplicticy we will let $\epsilon_0 t$ and $\epsilon t$ refer to the same agent. The derivation rules of Fig.2 is then replaced by the rules in Fig.3.

$$\frac{}{\epsilon_n t \xrightarrow{1} \epsilon_{n+1} t} \text{ (Wait)} \quad \text{and} \quad \frac{t \xrightarrow{a} t'}{\epsilon_n t \xrightarrow{a} t'} \text{ (Fulfill).}$$

**Fig.3.** Derivation rules for annotated finite delay

The *position* of a subagent is formalised as follows.

**Definition 8.** *Define* $Pos = \{1, 2\}^*$, *a set of* positions*, and let* $nil \in Pos$ *denote the empty sequence (the top position). Any term* $t$ *in* $\mathcal{T}_\epsilon$ *define a partial function* $t\colon Pos \rightharpoonup \mathcal{T}_\epsilon$*, given inductively (in the length of the position and the structure of* $t$*) by*

$$t(nil) = \begin{cases} t & \text{if } t \equiv a{:}t',\ t \equiv \mathsf{rec}\, x.t',\ t \equiv \Sigma_{i \in I} t_i \text{ or } t \equiv \epsilon t' \text{ for some } t', \\ t'(nil) & \text{if } t \equiv t' \lceil A, \\ undef & \text{otherwise}, \end{cases}$$

$$t(ip) = \begin{cases} t_i(p) & \text{if } t \equiv t_1 \times t_2, \\ t'(ip) & \text{if } t \equiv t' \lceil A, \\ undef & \text{otherwise}. \end{cases}$$

*For* $p \in Pos$ *and* $t$ *an annotated term, we will say that* $t(p)$ *is* waiting *if* $t(p) = \epsilon_n t'$ *for some term* $t'$ *and* $n > 1$.

Now, we can define when an infinite computation is inadmissible.

**Definition 9.** *An infinite computation* $t_0 \stackrel{\alpha_0}{\to} t_1 \stackrel{\alpha_1}{\to} t_2 \stackrel{\alpha_2}{\to} \ldots$ *derivable by the rules in Fig.1 and Fig.3 is* inadmissible *if and only if there exist* $j \in \omega$ *and a position* $p \in \{1,2\}^*$ *such that* $\forall j' \geq j$, $t_{j'}(p)$ *is waiting. We say that a computation is* admissible *if it is not inadmissible.*

It is not difficult to verify that a computation is inadmissible by the definition above if and only if it has a suffix with a waiting subagent which continues to wait forever, so the definition of admissibility coincides with that of [24] which we briefly gave in the beginning of the section.

The derivation transition systems for terms in $\mathcal{T}_\epsilon$ are generalised transition systems with the set of admissible computations given by Def. 9 above.

**Definition 10.** *Let* $t$ *be a term in* $\mathcal{T}_\epsilon$. *Then the* derivation transition system *for* $t$ *is the reachable* generalised *transition system* $\mathcal{O}_\epsilon(t) = (S, t, \to_t, Adm, Act)$, *where* $S = \{t' \mid t \to^* t'\}$, $\to_t = \to \cap \subseteq S \times Act \times S$ *is the relation defined by the rules in Fig.1 and Fig.3 restricted to states in S, and* $Adm \subseteq Comp\big((S, t, \to_t, Act)\big)$ *is the set of admissible computations as defined in Def. 9.*

*Remark 1.* We do not need to record *exactly* how many steps a delay has waited, just if has waited zero, one or more than one step continuously. This means that we could replace the first rule in Fig.3 by the rule $\epsilon_n t \stackrel{1}{\to} \epsilon_{\min\{n+1,2\}} t$ and only allow the numbers 0, 1 and 2 in annotations. The latter set of rules has the benefit of not giving rise to infinite graphs just because of the presence of a finite delay, which e.g. could be relevant for model and equivalence checking. These questions are outside the scope of the present paper.

# 6 Presheaf Semantics

In this section we will see that the category of separated presheaves $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ is well suited to give denotational semantics to SCCS$\epsilon$.

## 6.1 Semantics of Basic Operators

The denotation of sum is simply given by the coproduct in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$. The denotations of the remaining basic operators, restriction, action prefix, and synchronous product, can be obtained from the underlying functions on sequences using the free extension $(-)_!$ described in Sec. 1, in the case where $\mathsf{Q} = \mathbf{Sp}(\widehat{\mathsf{Inf}})$.

For $A \subseteq Act$, the *restriction on sequences* $(-)\lceil A \colon \mathsf{Inf} \to \mathsf{Inf}_\perp$ maps a sequence $\alpha$ to the (possible empty) sequence $\alpha' \leq \alpha$ being the longest prefix of $\alpha$ in $A^*$, i.e. the sequence $\alpha' \leq \alpha$ such that if $\alpha = \alpha' a \alpha''$ then $a \notin A$.

For $a \in Act$, the *action prefix on sequences* $a \colon \mathsf{Inf}_\perp \to \mathsf{Inf}$ maps a (possibly empty) sequence $\alpha$ to $a\alpha$.

The *synchronous product on sequences*, $\bullet \colon \mathsf{Inf} \times \mathsf{Inf} \to \mathsf{Inf}$ is the extension of the monoid product to sequences, i.e. for $\alpha, \beta \in \mathsf{Inf}$, $\alpha \bullet \beta = \gamma$, where $\gamma$ is the unique sequence such that $|\gamma| = \min\{|\alpha|, |\beta|\}$ and $\gamma_i = \alpha_i \bullet \beta_i$.

It is easy to see that the above mappings are monotone, and thus functors between partial order categories. By (implicitly) composing with the embeddings $\mathcal{Y}^{\circ}_{\mathsf{Inf}_\perp} : \mathsf{Inf}_\perp \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\mathcal{Y}_{\mathsf{Inf}} : \mathsf{Inf} \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}})$, we get functors $(-)\lceil A : \mathsf{Inf} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$, $a : \mathsf{Inf}_\perp \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\bullet : \mathsf{Inf} \times \mathsf{Inf} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$. Applying the extension $(-)_!$ we get the following denotations of basic operators.

**Basic operators:** For closed terms $t$, $t'$ and $t_i$, define

$$\mathcal{I}[\![\Sigma_{i\in I} t_i]\!] = \Sigma_{i\in I}\mathcal{I}[\![t_i]\!], \tag{6}$$

$$\mathcal{I}[\![a\!:\!t]\!] = \mathsf{sp}(a_! \circ \lfloor \mathcal{I}[\![t]\!] \rfloor), \tag{7}$$

$$\mathcal{I}[\![t \times t']\!] = \mathsf{sp}(\mathcal{I}[\![t]\!](\bullet_! \circ w)\mathcal{I}[\![t']\!]), \tag{8}$$

$$\mathcal{I}[\![t\lceil A]\!] = \mathcal{I}[\![t]\!]\lceil A_!, \tag{9}$$

where $a_! : \widehat{\mathsf{Inf}_\perp} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ is precomposed with the obvious lifting functor $\lfloor - \rfloor : \widehat{\mathsf{Inf}} \hookrightarrow \widehat{\mathsf{Inf}_\perp}$ and $\bullet_! : \widehat{\mathsf{Inf} \times \mathsf{Inf}} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ is precomposed with the (connected colimit-preserving [7]) functor $w : \widehat{\mathsf{Inf}} \times \widehat{\mathsf{Inf}} \to \widehat{\mathsf{Inf} \times \mathsf{Inf}}$ defined (on objects) by $w(X, Y)(\alpha, \beta) = X(\alpha) \times Y(\beta)$. The semantic functions are extended in the obvious way to terms $t$ with free variables in a set $\mathcal{V}$, yielding functors

$$\mathcal{I}[\![t]\!]_{\mathcal{V}} : \prod_{x\in\mathcal{V}} \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}}).$$

Since the functors are build up from connected colimit preserving functors it follows that they themselves preserve connected colimits.

The first three definitions (6)-(8) above only give the denotation up to isomorphism. It is helpful, e.g. in showing correspondence with the operational semantics, to give an explicit semantics $[\![t]\!]$ such that $[\![t]\!] \cong \mathcal{I}[\![t]\!]$. We will just give the action on objects. The tags $\mathsf{sum}$ and $\times$ are used to indicate clearly how an element came about, which we will use in App. B.

$$[\![t\lceil A]\!]\alpha = \{e \mid \alpha \in A^\infty \text{ and } e \in [\![t]\!]\alpha\}. \tag{10}$$

$$[\![\Sigma_{i\in I} t_i]\!]\alpha = \big\{(\mathsf{sum}\ i, (\alpha, e)) \mid i \in I \text{ and } e \in [\![t_i]\!]\alpha\big\}. \tag{11}$$

$$[\![a\!:\!t]\!]\alpha = \begin{cases} \lfloor [\![t]\!] \rfloor \alpha' & \text{if } \alpha = a\alpha', \\ \varnothing & \text{otherwise,} \end{cases} \tag{12}$$

where we choose to represent $\lfloor - \rfloor : \widehat{\mathsf{Inf}} \hookrightarrow \widehat{\mathsf{Inf}_\perp}$ explicitly by

$$\lfloor X \rfloor \alpha = \begin{cases} \{*\} & \text{if } \alpha = \perp, \\ X\alpha & \text{otherwise.} \end{cases} \tag{13}$$

$$\begin{aligned}[\![t_1 \times t_2]\!]\alpha = \{&(\beta, e_1) \times (\gamma, e_2) \mid \\ &\beta, \gamma \in \mathsf{Inf}.\beta \bullet \gamma = \alpha \text{ and } e_1 \in [\![t_1]\!]\beta \text{ and } e_2 \in [\![t_2]\!]\gamma\}.\end{aligned} \tag{14}$$

### 6.2 Semantics of Recursion

For *recursion* we need to take care. In a “standard” semantics one would take least fixed points, i.e. initial algebras as the meanings of recursion. However in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$, this would not reflect that it is admissible to unfold a recursion infinitely. An explicit example that illustrates this is given below, showing that the initial algebra of the functor corresponding to the delay equation given in Sec. 2 will be the proper denotation of *finite* delay and not the delay operator derived using recursion. The solution is to take *final* co-algebras as the meanings of recursion.

**Infinite recursion:** For a term $t$ with one free variable $x$, define

$$\mathcal{I}[\![\mathsf{rec}\, x.t]\!] = \nu \mathcal{I}[\![t]\!],$$

i.e. (the object of) a final co-algebra of the endofunctor $\mathcal{I}[\![t]\!] \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$. For this to be well defined, we must show existence of final co-algebras for all functors. We will use Lem. 2 given in Sec. 1 to construct final co-algebras for all relevant endofunctors as limits of $\omega^{op}$-chains. The definition is then extended to processes with more than one variable in the usual way as a limit with parameters [20]. From the explicit definitions given in Eq. (10)-(14) we can show that all basic operators preserve $\omega^{op}$-limits. From the general fact that limits commute with limits [20] we get that recursion preserves $\omega^{op}$-limits as well, i.e. if $\mathsf{rec}\, x.t$ has free variables then $\mathcal{I}[\![\mathsf{rec}\, x.t]\!]$ preserves $\omega^{op}$-limits.

**Lemma 7.** *Let $t$ be a (possibly open) term of SCCS with free variables in $\mathcal{V}$. If*

$$\mathcal{I}[\![t]\!]_{\mathcal{V}} \colon \prod_{x \in \mathcal{V}} \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$$

*(is well defined and) preserves $\omega^{op}$-limits then*

$$\mathcal{I}[\![t \lceil A]\!]_{\mathcal{V}} \colon \prod_{x \in \mathcal{V}} \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$$

*(is well defined and) preserves $\omega^{op}$-limits, and similarly for sum, prefix, synchronous product and recursion.*

As for the basic operators, we can give an explicit denotation of recursion $[\![\mathsf{rec}\, x.t]\!] \cong \mathcal{I}[\![\mathsf{rec}\, x.t]\!]$. First we choose an explicit representation of a final presheaf $\top$ by definining $\top\alpha = \{*\}$. Now we use the explicit definition of limits in the category $\mathsf{Set}$ to define

$$[\![\mathsf{rec}\, x.t]\!]\alpha = \big\{ \langle e_0, e_1, \dots, e_n, \dots \rangle \in \prod_{n \in \omega} [\![t]\!]^n(\top)\alpha \mid [\![t]\!]^n(\tau)_\alpha e_{n+1} = e_n \big\}, \qquad (15)$$

where $\tau \colon [\![t]\!](\top) \to \top$ is the natural transformation given by $\tau_\alpha(e) = *$ for any $e \in [\![t]\!](\top)\alpha$. We have projections $\pi_n \colon [\![\mathsf{rec}\, x.t]\!] \to [\![t]\!]^n(\top)$ and by universality we get an (explicit) isomorphism $\rho_t \colon [\![\mathsf{rec}\, x.t]\!] \to [\![t]\!]([\![\mathsf{rec}\, x.t]\!])$, such that

$$\begin{tikzcd} {[\![\mathsf{rec}\, x.t]\!]} \arrow[r, "\pi_{n+1}"] \arrow[d, "\rho_t"'] & {[\![t]\!]^{n+1}(\top)} \\ {[\![t]\!]([\![\mathsf{rec}\, x.t]\!])} \arrow[ur, "{[\![t]\!](\pi_n)}"'] & \end{tikzcd} \qquad (16)$$

commutes for any $n \in \omega$. Note that, in general if $t$ has free variables $\mathcal{V} \uplus \{x\}$ then $\rho_t$ and $\pi_n$ are natural transformations.

We have now given semantics to all operators in SCCS$\epsilon$ except for finite delay. It is worth remarking, that already at this stage it is clear that this semantics will not (in general) correspond to the operational semantics given in Sec. 5. A simple example showing this is provided by the (disastrous) term $\mathsf{rec}\,x.x$. According to the operational semantics, this term denotes the process that cannot perfom any actions, which is also the process denoted by the empty sum $0$. It is not diffucult to compute the appropriate limit finding that $\mathcal{I}[\![\mathsf{rec}\,x.x]\!] \cong \top$, i.e. (the) final object in $\widehat{\mathsf{Inf}}$, which in no sensible way can be equated to the denotation of the empty sum, which is the *initial* object in $\widehat{\mathsf{Inf}}$. (Note that this is indeed the result if one constructs the initial algebra instead).

However, as we will see below, we get the desired correspondence if we restrict the language to only allow *guarded* recursion.

### 6.3 Semantics of Finite Delay

As mentioned above, the denotation of finite delay comes about as the *initial* algebra of the functor corresponding to the delay equation.
**Finite delay:** For a closed term $t$, define

$$\mathcal{I}[\![\epsilon t]\!] = \mu \mathcal{I}[\![1\!:\!x + t]\!],$$

i.e. (the object of) an initial algebra of the endofunctor $\mathcal{I}[\![1\!:\!x{+}t]\!]\colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$. This initial algebra exists by Lem. 3 since the denotation of prefixing preserves connected colimits and the denotation of sum all colimits. The definition is extended to open terms (in which $t$ is not free) as a colimit with parameters.

From the explicit definition of colimits in $\mathsf{Set}$, we find that we can take

$$[\![\epsilon t]\!]\alpha = \big\{ \big(\mathsf{del}\ n, (\alpha', e)\big) \mid n \in \omega, \alpha = 1^n\alpha' \text{ and } e \in \lfloor[\![t]\!]\rfloor\alpha' \big\} \tag{17}$$

as explicit definition of finite delay on objects (again the tag $\mathsf{del}$ is used to indicate clearly that the element arise from the denotation of a finite delay). For $\beta \leq \alpha$, define $[\![\epsilon t]\!]([\alpha, \beta])$ by

$$\big(\mathsf{del}\ n, (\alpha', e)\big) \cdot [\beta, \alpha] = \begin{cases} \big(\mathsf{del}\ n, (\beta', e \cdot [\beta', \alpha'])\big) & \text{if } \beta = 1^n\beta', \\ \big(\mathsf{del}\ m, (\bot, *)\big) & \text{if } \beta = 1^m \text{ for } m < n, \end{cases}$$

for $n \in \omega$, $\alpha = 1^n\alpha'$ and $e \in \lfloor[\![t]\!]\rfloor\alpha'$.

To guarantee that the denotation of recursion is still well-defined, we need to check that the denotations of finite delay preserve $\omega^{op}$-limits. This can be done from the explicit definition given above.

**Lemma 8.** *Let $t$ be a (possibly open) term of SCCS$\epsilon$ with free variables in $\mathcal{V}$. If*

$$\mathcal{I}[\![t]\!]_{\mathcal{V}}\colon \prod_{x \in \mathcal{V}} \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$$

*(is well defined and) preserves $\omega^{op}$-limits then*

$$\mathcal{I}[\![\epsilon t]\!]_{\mathcal{V}} : \prod_{x \in \mathcal{V}} \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$$

*(is well defined and) preserves $\omega^{op}$-limits.*

This completes the definition of our denotational semantics of SCCS$\epsilon$ in the category of seperated presheaves $\mathbf{Sp}(\widehat{\mathsf{Inf}})$.

### 6.4 Extended Bisimulation Congruence

From the fact that the denotations (in $\widehat{\mathsf{Inf}}$) of all basic operators are built from connected colimit preserving functors, it follows that they preserve open maps in $\widehat{\mathsf{Inf}}$. Using the fact that the inclusion of $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ in $\widehat{\mathsf{Inf}}$ is full, together with proposition 5 in [19] we get that this holds in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ as well. It is easy to show from the explicit definition that the denotations of finite delay preserve open maps as well (alternatively one could use the same technique as used in [7] for showing that denotations of recursions (given by initial algebras) preserve open maps).

**Proposition 7.** *Extended bisimulation is a congruence with respect to all basic operators of SCCS$\epsilon$ as well as finite delay.*

However, when it comes to *recursion* we meet a problem: What is the "right" notion of bisimulation (from open maps) for denotations of open terms, i.e. functors between presheaf categories? In [7] the notion of open maps is extended to open natural transformations, being natural transformations for which all components are open maps. This is shown to be sufficient to garentee that open map bisimulation is a congruence with respect to the denotations of recursion (given by *initial* algebras) in a CCS-like calculus. In [29, 4] is suggested a slightly stronger notion of open maps between connected-colimit preserving functors between presheaf categories, which can be regarded as objects of a presheaf category and thus comes with a canonical notion of open maps. However, to apply this notion of bisimulation all functors must be connected-colimit preserving functors, which is not known to be the case in our setting (because of the use of final co-algebras). The notion of open natural transformations could be used, but it remains to be shown if this is sufficient to give the desired congruence property.

### 6.5 Full Abstraction

Using the representation theorem in Sec. 3 we can express the denotational semantics given above in terms of generalised synchronisation trees, defining $\mathcal{D}_\epsilon(t) = \mathcal{E}l([\![t]\!])$. This allows us to relate the denotational semantics directly to the operational semantics given in Sec. 5 within the category $\mathsf{GTS}$. First of all we will restrict attention to terms with only *guarded recursion*. Recall from e.g. [25] that a recursion $\mathsf{rec}\, x.t$ is guarded, if all free occurences of $x$ in $t$ is guarded, that is, within a subterm $a : t'$ of $t$ for some action $a \in Act$. Let $\mathcal{T}_g$ refer to the set of all closed, possibly annotated terms of SCCS$\epsilon$ with only guarded recursion. We will say that a term $t$ in $\mathcal{T}_g$ is *standard* if for all subterms $e_n t'$ it holds that $n = 0$. We will then show, that if we quotient by open map bisimulation, the denotational semantics for standard terms in $\mathcal{T}_g$ is in fact *equationally fully abstract* with respect to extended bisimulation. This means that for any two standard terms $t$ and $t'$ of $\mathcal{T}_g$, the presheaves $[\![t]\!]$ and $[\![t']\!]$ are bisimilar if and only if the generalised transition systems $\mathcal{O}_\epsilon(t)$ and $\mathcal{O}_\epsilon(t')$ arising from the operational semantics are extended bisimilar. As remarked in Sec. 6.2 above, we cannot obtain this result for all terms of SCCS$\epsilon$.

The proof (see App. B for a more detailed proof outline) goes by showing that there exists an $\mathsf{Inf}_\perp$-open morphism of generalised transition systems from $\mathcal{D}_\epsilon(t)$ to $\mathcal{O}_\epsilon(t)$ for any term $t$ in $\mathcal{T}_g$.

**Proposition 8.** *Let $t$ be a standard term in $\mathcal{T}_g$. Then there exists an $\mathsf{Inf}_\perp$-open morphism of generalised transition systems $F_t\colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$.*

From the proposition above and Prop. 6 and Cor. 1 in Sec. 4 we can now deduce the desired result.

**Theorem 3.** *Let $t$ and $t'$ be terms in $\mathcal{T}_g$. Then $[\![t]\!]$ and $[\![t']\!]$ are open map bisimilar if and only if $\mathcal{O}_\epsilon(t)$ and $\mathcal{O}_\epsilon(t')$ are extended bisimilar.*

## 7 Conclusion and Future Work

This paper has two main contributions. The first is a generalisation of the categorical models for concurrency as developed in [30, 19, 4], providing both a generalised transition system and a presheaf model for *infinite* computations, suitable for agents with a notion of *fairness* or *inadmissible* infinite computations. The generalised transition systems are instances of those proposed in [14] and the *extended bisimulation* given there is shown to coincide with the abstract bisimulation from span of open maps in our model. The second main contribution is that we give both an operational semantics and a denotational semantics for SCCS with finite delay, representing the notion of inadmissible infinite computations precisely as the operational semantics in [24] and allowing behaviours to be discriminated up to *extended bisimulation*. This notion of bisimulation is a strictly finer, and as argued in the present paper and in [1], more intuitive, equivalence than the one obtained from the fortification preorder in [24], which except for [1] has been the basis for previous denotational semantics of SCCS with finite delay [13, 18, 17]. Benefitting from the categorical presentation, our semantics appears to give a conceptually simpler treatment of infinite computations than the one in [1]. If one restricts attention to agents for which products and restrictions are disallowed within recursions and change the operational semantics according to the remark in Sec. 5 all agents will be assigned *finite* (generalised) transition systems. Being essentially CTL$^*$-bisimulation one gets decidability if the characterisation of admissible infinite computations can be described by a Muller condition, which seems not to be difficult.

A number of questions remains to be explored. An obvious question is if one could generalise the finite delay to a *fair recursion* as in [17]. Work is in progress on a notion of open maps between denotations of open terms stronger than the one in [7], for which open map bisimulation is hopefully a congruence with respect to recursion. We get a characteristic HML-like path logic [19] for extended bisimulation from the open maps approach, which should be compared to the characteristic logic given in [14] and an edge-labelled version of the (fair) CTL$^*$ logic. It would be interesting to explore if there is any relationship between the present approach and the more traditional domain theoretical approach to fairness and countable non-determinism as in e.g. [27]. Finally, we hope to be able to extend the presheaf model for (finitary) dataflow given in [16] to infinite computations along the lines of the present paper, giving a model of dataflow in which fairness, and in particular *fair merge* [26], can be expressed.

**Acknowledgements:** Thanks to Glynn Winskel, Marcelo Fiore and Prakash Panangaden for helpful and encouraging discussions.

## References


1. P. Aczel. A semantic universe for fairness. Preliminary Draft, 1996.
2. A. Aziz, V. Singhal, and F. Balarin. Equivalences for fair kripke structures. In S. Abiteboul and E. Shamir, editors, *ICALP '94: International Colloquium on Automata, Languages, and Programming*, volume 820 of *LNCS*, pages 364–375. Springer-Verlag, 1994.
3. Barr and Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.
4. G. L. Cattani. *Presheaf models for concurrency*. PhD thesis, Aarhus University, 1999.
5. G. L. Cattani, A. J. Power, and G. Winskel. A categorical axiomatics for bisimulation. In *Proceedings of the 9th International Conference on Concurrency Theory, CONCUR '98*, volume 1466 of *LNCS*, pages 581–596. Springer-Verlag, 1998.
6. G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the pi-calculu. In *CTCS'97*, volume 1290 of *LNCS*, pages 106–126. Springer, 1997.
7. G. L. Cattani and G. Winskel. Presheaf models for concurrency. In *CSL'96*, volume 1258 of *LNCS*, pages 58–75. Springer, 1997.
8. G. L. Cattani and G. Winskel. Presheaf models for ccs-like languages. *Submitted*, 1999.
9. A. Cheng and M. Nielsen. Open maps (at) work. Research Series RS-95-23, BRICS, Department of Computer Science, University of Aarhus, Apr. 1995.
10. E. A. Emerson. Temporal and modal logic. In J. van Leeuven, editor, *Handbook of Theoretical Computer Science*, volume 2, chapter 16, pages 996–1072. Elsevier Science B.V., 1990.
11. E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the Association of Computing Machinery*, 33(1):151–178, January 1986.
12. M. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps. In *Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science*, 1999.
13. M. Hennessy. Modelling finite delay operators. Technical report, University of Edinburgh, 1983.
14. M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, pages 23–52, 1985.
15. T. T. Hildebrandt. *Categorical Models for Concurrency: Independence, Fairness and Dataflow*. PhD thesis, Department of Computer Scence, University of Aarhus, Denmark, October 1999.
16. T. T. Hildebrandt, P. Panangaden, and G. Winskel. A relational model of non-deterministic dataflow. In *CONCUR'98*, volume 1466 of *LNCS*, pages 613–628. Springer-Verlag, 1998.
17. M. Huth and M. Kwiatkowska. The semantics of fair recursion with divergence. Submitted. Technical report CSR-96-7.

18. M. Huth and M. Kwiatkowska. Finite but unbounded delay in synchronous CCS. In A. Edalat, S. Jourdan, and G. McCusker, editors, *Advanced methods in theory and formal methods of computing: Proceedings of the third Imperial College workshop April 1996*, pages 312–323. Imperial College Press, 1996.
19. A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *LICS '93 special issue of Information and Computation*, 127(2):164–185, juni 1996. Available as BRICS report, RS-94-7.
20. S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.
21. S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 1992.
22. J. Malitz. *Introduction to Mathematical Logic*. Undergraduate Texts in Mathematics. Springer-Verlag, 1979.
23. R. Milner. A calculus of communicating systems, 1980.
24. R. Milner. A finite delay operator in synchronous ccs. Technical report, University of Edinburgh, Dept. of Computer Science, Kings Buildings, 1982.
25. R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.
26. P. Panangaden. The expressive power of indeterminate primitives in asynchronous computations. Technical Report SOCS-95.8, School of CS, McGill, 1995.
27. G. Plotkin. A powerdomain for countable non-determinism. In *Automata, Languages and Programming (ICALP), Ninth Colloquium*, volume 140 of *LNCS*, pages 418–428. Springer-Verlag, 1982.
28. G. Winskel. Generalised synchronisation trees. Handwritten notes, 1983.
29. G. Winskel. A linear metalanguage for concurrency. In *AMAST '98*, volume 1548 of *LNCS*, pages 42–58. Springer-Verlag, 1998.
30. G. Winskel and M. Nielsen. *Handbook of Logic in Computer Science*, volume IV, chapter Models for concurrency. OUP, 1995.
31. G. Winskel and M. Nielsen. Presheaves as transition systems. In D. Peled, V. Pratt, and G. Holzmann, editors, *POMIV'96*, volume 29 of *DIMACS*. AMS, july 1996.
32. O. Wyler. *Lecture Notes on Topoi and Quasitopoi*. World Scientific, 1991.

## A Grothendieck topology for a partial order

Here we give the definitions from [21] of a *Grothendieck topology* for a category $\mathsf{P}$ and the *sup* topology, specialised to the case where $\mathsf{P}$ is a partial order. Let $P$ be a partial order and $p \in P$. Define $p{\downarrow} = \{p' \in P \mid p' \leq p\}$. A *sieve* $S$ on $p$ is then a set $S \subseteq p{\downarrow}$, i.e. a downwards closed set below $p$.

**Definition 11 (Grothendieck topology for a partial order).** *A* Grothendieck topology *for a partial order* $\mathsf{P}$*, is a function $J$ which assigns to each object $p$ of* $\mathsf{P}$ *a set $J(p)$ of sieves on $p$, in such a way that*

C1*: (maximal sieve) $p{\downarrow} \in J(p)$,*
C2*: (stability) if $S \in J(p)$ and $q \leq p$ then $q{\downarrow} \cap S \in J(q)$,*
C3*: (transitivity) if $S \in J(p)$ and $R$ is any sieve on $p$ such that $q{\downarrow} \cap R \in J(q)$ for all $q \in S$, then $R \in J(p)$.*

Assume $J$ is a topology for a partial order $P$. We will now describe when a presheaf $X\colon \mathsf{P}^{\mathsf{op}} \to \mathsf{Set}$ in $\widehat{\mathsf{P}}$ is a sheaf with respect to $J$. Assume $p$ is an element of $P$ and $S \in J(p)$, i.e. a *sieve covering p*. A *matching family* for $S$ of elements of $X$ is a function that assigns to each element $q \in S$ an element $x_q \in X(q)$ such that $x_q \cdot [r,q] = x_r$ for any $r \leq q$. Given such a matching family, an element $x \in X(p)$ is an *amalgation*, if $x \cdot [q,p] = x_q$ for all $q \in S$. Then $X$ is respectively a *separated presheaf* or a *sheaf* with respect to $J$ if for any object $p \in P$, any matching family for any sieve $S \in J(p)$ has respectively *at most one* or *a unique* amalgation.

**Definition 12 (separated presheaves and sheaves).** *For a partial order* $\mathsf{P}$ *and a Grothendieck topology* $J$ *on* $\mathsf{P}$*, let* $\mathbf{Sp}_J(\widehat{\mathsf{P}})$ *and* $\mathbf{Sh}_J(\widehat{\mathsf{P}})$ *be the full subcategories of* $\widehat{\mathsf{P}}$ *induced by respectively the* separated presheaves *and the* sheaves *with respect to* $J$*. If the topology* $J$ *is clear from the context, we will just write respectively* $\mathbf{Sp}(\widehat{\mathsf{P}})$ *and* $\mathbf{Sh}(\widehat{\mathsf{P}})$.

For a sequence $\alpha$ in $\mathsf{Inf}$ (as defined in Sec. 1), a sieve on $\alpha$ is simply a prefix closed set of sequences below $\alpha$. We only use the *sup* topology on $\mathsf{Inf}$, which to each sequence $\alpha$ assigns the set $\{S \mid S \text{ is a sieve on } \alpha \text{ and } \bigsqcup S = \alpha\}$, i.e. of all sieves that have $\alpha$ as supremum. It is easy to check that this satisfy the conditions in Def. 11, and that it works for any partial order. This topology is in fact the *canonical* topology for $\mathsf{Inf}$, being the largest topology such that $\mathcal{Y}_{\mathsf{Inf}}\alpha$ is a sheaf for any $\alpha$.

**Definition 13 (sup topology for $\mathsf{Inf}$).** *For the partial order* $\mathsf{Inf}$*, the* sup topology $J$ *is given by* $J(\alpha) = \{\alpha\!\downarrow, \{\beta \mid \beta \leq_f \alpha\}\}$*, for* $\alpha \in \mathsf{Inf}$

Note that if $\alpha$ is finite then $J(\alpha)$ contains just $\alpha\!\downarrow$, i.e. the maximal sieve on $\alpha$.

## B Proof of Full Abstraction

We will here give a more detailed proof outline for Prop. 8 of Sec. 6.5 as repeated below. Recall that $\mathcal{T}_g$ refer to the set of all closed terms of SCCS$\epsilon$ with only guarded recursion and that a term is standard if for all subterms $\epsilon_n u''$, $n = 0$. Let $\mathcal{T}_g^o$ refer to the set of, possible open, terms of SCCS$\epsilon$ with only guarded recursion.

**Proposition 9 (Prop. 8 of Sec. 6.5).** *Let* $t$ *be a standard term in* $\mathcal{T}_g$*. Then there exists an* $\mathsf{Inf}_\perp$*-open morphism of generalised transition systems* $F_t\colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$.

We will need some preliminary definitions. For $t$ a term in SCCS$\epsilon$, $FV(t)$ will denote the set of free variables in $t$. As in [24] [3] we define $gd(t)$, the *guard-depth* of $t$ by

- $gd(x) = gd(a\!:\!t) = 0$,
- $gd(\Sigma_{i\in I} t_i) = \sup\{gd(t_i) + 1 \mid i \in I\}$,
- $gd(t_1 \times t_2) = \max\{gd(t_1) + 1, gd(t_2) + 1\}$, and
- $gd(\mathsf{rec}\, x.t) = gd(t\lceil A) = gd(\epsilon t) = gd(t) + 1$.

[3] However, we use the convention from [22] that $\lambda + 1$ is the successor of $\lambda$.

This is a well defined ordinal, but not necessarily a finite number because sums can be infinite. As in [24] the following is a key property of $gd$ for use in inductive proofs in the guard depth of terms with only guarded induction.

**Lemma 9.** *If $x$ is guarded in $t$ then $gd(t[t'/x]) = gd(t)$.*

*Proof.* By a straightforward structural induction.

For a term $t$ in $\mathcal{T}_\epsilon$ we define $sd(t)$, the *subagent depth* of $t$ by

- $sd(a\!:\!t) = sd(\Sigma_{i\in I} t_i) = sd(\mathsf{rec}\, x.t) = sd(\epsilon t) = 0$,
- $sd(t_1 \times t_2) = 1 + \max\{sd(t_1), sd(t_2)\}$, and
- $sd(t\lceil A) = 1 + sd(t)$.

This is simply the maximal depth of a subagent and thus always finite.

For a gst $T = (S, i, \to, Adm, Act)$ and $s \in S$ we define *the gst above $s$ in $T$* by $T_{s\triangleleft} = (S_{s\triangleleft}, s, \to_{s\triangleleft}, Adm_{s\triangleleft}, Act)$, where

- $S_{s\triangleleft} = \{s' \mid s \to^* s'\}$,
- $\to_{s\triangleleft} = \to \cap\, (S_{s\triangleleft} \times Act \times S_{s\triangleleft})$ and
- $Adm_{s\triangleleft} = Adm \cap \to^{\infty}_{s\triangleleft}$.

For any term $t$ in $\mathcal{T}_\epsilon$, let $\mathcal{D}_\epsilon(t) = (S_{d(t)}, (\perp, *), \to_t, Adm_{d(t)}, Act)$. Recall that $S_{d(t)} = \big\{(\alpha, e) \mid \alpha \in \mathsf{Inf} \text{ and } e \in \lfloor\![\![t]\!]\!\rfloor(\alpha)\big\}$ and $*$ is the unique element of $\lfloor\![\![t]\!]\!\rfloor(\perp)$. Let $\mathcal{O}_\epsilon(t) = (S_{o(t)}, t, \to, Adm_{o(t)}, Act)$. Note that if $t'$ is a closed term and $t$ is a term with one free variable, say $x$, then $[\![t[t'/x]]\!] = [\![t]\!]([\![t']\!])$. For $t$ a term in $\mathcal{T}_g$ and $s = (\alpha, e) \in S_{d(t)}$ define *the height of $s$* by $h(s) = |\alpha| \in \omega$. Note that if $h(s) = n$ then $(\perp, *) \to^n s$.

We are now ready to define the underlying maps of states $f_t\colon S_{d(t)} \to S_{o(t)}$ for the morphisms $F_t\colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$.

**Definition 14.** *Let $\mathcal{S}_\mathcal{T} = \{(s,t) \mid s \in S_{d(t)} \text{ and } t \in \mathcal{T}_g\}$. Define $f\colon \mathcal{S}_\mathcal{T} \to \mathcal{T}_g$ by well founded recursion as follows (writing $f_t(s)$ for $f(s,t)$)*

- $f_t(\perp, *) = t$,
- $f_{a:t}(a\alpha, e) = f_t(\alpha, e)$,
- $f_{\Sigma_{i\in I} t_i}\big(\alpha, (\mathsf{sum}\ i, s)\big) = f_{t_i}(s)$,
- $f_{t_1 \times t_2}(\alpha, s_1 \times s_2) = f_{t_1}(s_1) \times f_{t_2}(s_2)$,
- $f_{\mathsf{rec}\, x.t}(s) = f_{t[\mathsf{rec}\, x.t/x]}\big(\mathcal{E}l(\rho_t)s\big)$ *if* $h(s) > 0$,
- $f_{t\lceil A}(s) = f_t(s)\lceil A$ *if* $h(s) > 0$,
- $f_{\epsilon_n t}\big(1^{n'}, (\mathsf{del}\ n', (\perp, *))\big) = \epsilon_{n+n'} t$,
- $f_{\epsilon_n t}\big(1^{n'}\alpha, (\mathsf{del}\ n', s)\big) = f_t(s)$ *if* $|\alpha| > 0$.

*where $\rho_t\colon [\![\mathsf{rec}\, x.t]\!] \to [\![t]\!]([\![\mathsf{rec}\, x.t]\!])$ is the isomorphism defined in Sec.6.2 and the well founded order on $\mathcal{S}_\mathcal{T}$ is the lexicographical order given by $(s_1, t_1) < (s_2, t_2)$ if $h(s_1) < h(s_2)$ or $h(s_1) = h(s_2)$ and $gd(t_1) < gd(t_2)$.*

It is not difficult to check from the definitions in Sec. 6 that $f_t$ is only applied to states in $S_{d(t)}$ on the right hand side of the defining equations above.

From the map $f\colon \mathcal{S}_\mathcal{T} \to \mathcal{T}_g$ we get a collection of maps $\{f_t\colon S_{d(t)} \to \mathcal{T}_g \mid t \in \mathcal{T}_g\}$ that are nicely related to each other.

**Lemma 10.** *Let $\{f_t \colon S_{d(t)} \to \mathcal{T}_g \mid t \in \mathcal{T}_g\}$ be the collection of maps given above. Then there exists a collection of isomorphisms of generalised synchronisation trees $\{\sigma_{t,s} \colon \mathcal{D}_\epsilon(t)_{s\triangleleft} \to \mathcal{D}_\epsilon\big(f_t(s)\big) \mid t \in \mathcal{T}_g$ and $s \in S_{d(t)}\}$ such that if $s \to_t^* s'$ in $\mathcal{D}_\epsilon(t)$ then*

$$(f_t(s) = t') \Rightarrow f_t(s') = f_{t'}\big(\sigma_{t,s}(s')\big), \tag{18}$$

*Proof.* (Sketch) We proceed by induction in the height of the states $s$. First we define $\sigma_{t,s} \colon \mathcal{D}_\epsilon(t)_{s\triangleleft} \to \mathcal{D}_\epsilon\big(f_t(s)\big)$ for $t \in \mathcal{T}_g$ and $s = (\perp, *) \in S_{d(t)}$, i.e. for all roots. Then $\mathcal{D}_\epsilon(t)_{s\triangleleft} = \mathcal{D}_\epsilon(t)$ and $f_t(s) = t$ so we can define $\sigma_{t,s} = 1_{\mathcal{D}_\epsilon(t)}$. We then define $\sigma_{t,s} \colon \mathcal{D}_\epsilon(t)_{s\triangleleft} \to \mathcal{D}_\epsilon\big(f_t(s)\big)$ for $t \in \mathcal{T}_g$, $s \in S_{d(t)}$ and $h(s) = 1$ by transfinite induction in $gd(t)$. For the induction step, assume $t \in \mathcal{T}_g$, $s \in S_{d(t)}$ and $h(s) = n + 1$. Then there exists a unique $s_n$ such that $s_n \to_t s$ and $h(s_n) = n$. For $s \to_t^* s'$ define $\sigma_{t,s}(s') = \sigma_{f_t(s_n),\sigma_{t,s_n}(s)}(\sigma_{t,s_n}(s'))$. It is not difficult to verify that this indeed defines an isomorphism from $\mathcal{D}_\epsilon(t)_{s\triangleleft}$to $\mathcal{D}_\epsilon\big(f_t(s)\big)$. Assuming $f_t(s) = t'$ and $f_t(s_n) = t''$ we get by induction $f_{t''}\big(\sigma_{t,s_n}(s)\big) = t'$ and $f_t(s') = f_{t''}\big(\sigma_{t,s_n}(s')\big) = f_{t'}\big(\sigma_{t'',\sigma_{t,s_n}(s)}(\sigma_{t,s_n}(s'))\big) = f_{t'}\big(\sigma_{t,s}(s')\big)$.

From the lemma below it follows that the maps just defined are the underlying maps of $\mathsf{Fin}_\perp$-open morphisms from $fin(\mathcal{D}_\epsilon(t))$ to $fin(\mathcal{O}_\epsilon(t))$.

**Lemma 11.** *Let $\{f_t \colon S_{d(t)} \to \mathcal{T}_g \mid t \in \mathcal{T}_g\}$ be the collection of maps given in Def. 14 above. If $f_t(s_0) = t_0$ for $s_0 \in S_{d(t)}$ then*

$$\big(\exists s_1 \in S_{d(t)}.s_0 \xrightarrow{a}_t s_1 \textit{ and } f_t(s_1) = t_1\big) \textit{ if and only if } t_0 \xrightarrow{a} t_1 \ \ , \tag{19}$$

*where $\to$ is the transition relation given by the operational semantics in Fig. 1 and Fig.3.*

*Proof.* We first show by transfinite induction in $gd(t)$ that

$$\big(\exists s_1 \in S_{d(t)}.(\perp, *) \xrightarrow{a}_t s_1 \text{ and } f_t(s_1) = t_1\big) \text{ if and only if } t \xrightarrow{a} t_1 \ \ .$$

Then (19) follows for $s_0 \in S_{d(t)}$ and $f_t(s_0) = t_0$ by using (18) of Lem. 10.

**Corollary 2.** *The maps $f_t$ as given above defines for $t \in \mathcal{T}_g$ a map $f_t \colon S_{d(t)} \to S_{o(t)}$ which is the underlying map of a $\mathsf{Fin}_\perp$-open morphism from $fin(\mathcal{D}_\epsilon(t))$ to $fin(\mathcal{O}_\epsilon(t))$.*

To show that the maps $f_t$ define maps of *generalised* transition systems we show that they preserve admissible computations. For an infinite admissible computation $\phi$ of $\mathcal{D}_\epsilon(t)$ we can always find a non-empty prefix of the image of $\phi$ under $f_t$, in which all initially waiting subagents are fullfilled.

**Lemma 12.** *Let $t$ be a term in $\mathcal{T}_g$ and $\phi \in Adm_{d(t)} \cap \to^\omega$ an infinite admissible computation of $\mathcal{D}_\epsilon(t)$. Assume $\phi_n = (s_n, a_n, s_{n+1})$ for $n \in \omega$ and $f_t(s_n) = t_n$. Then there exists $n > 0$ such that*

$$\forall p \in Pos, \exists m \leq n.t_m(p) \textit{ is } \text{not } \textit{waiting.}$$

*Proof.* Easy induction in $sd(t_0)$ using Lem. 10.

It follows by a simple mathematical induction that $f_t$ preserves admissibility.

**Lemma 13.** *Let $t$ be a term in $\mathcal{T}_g$. Then $f_{t\infty}(Adm_{d(t)}) \subseteq Adm_{o(t)}$, where $f_{t\infty}$ is the extension of $f_t$ to computations defined as in Def. 5.*

We can now conclude from Lem. 4, Cor. 2 and Lem. 13 that $f_t$ defines a morphism of generalised transition systems.

**Proposition 10.** *Let $t$ be a term in $\mathcal{T}_g$. Then $f_t \colon S_{o(t)} \to S_{d(t)}$ is the underlying map of states of a morphism of generalised transition systems which we will refer to as $F_t \colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$.*

To show that $F_t \colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$ is an $\mathsf{Inf}_\perp$-*open* morphism we neet to check the two zig-zag conditions of Prop. 5 in Sec. 4. As already mentioned above, the first condition follows directly from Lem. 11. To show the second condition, it suffices to show that $f_t \colon S_{o(t)} \to S_{d(t)}$ *reflects* admissible computations, i.e. that $Adm_{f\,d(t)} \subseteq Adm_{d(t)}$, where $Adm_{f\,d(t)} = f_{t\infty}^{-1}(Adm_{o(t)}) = \{\phi \in Comp(\mathcal{D}_\epsilon(t)) \mid f_{t\infty}(\phi) \in Adm_{o(t)}\}$. The proof goes by structural induction in $t$ and for the case $t = \mathsf{rec}\, x.t'$ we will add a term $\top$ to the calculus SCCS$\epsilon$. The operational semantics is extended by adding the rule

$$\frac{}{\top \xrightarrow{a} \top} \quad (a \in Act).$$

As denotation of $\top$ we take the explicit terminal element of $\mathbf{Sp}(\widehat{\mathsf{Inf}})$, i.e. $[\![\top]\!]\alpha = \{*\}$. The map $f_\top \colon S_{d(\top)} \to S_{o(\top)}$ and isos $\sigma_{\top,s} \colon \mathcal{D}_\epsilon(\top)_{s\triangleleft} \to \mathcal{D}_\epsilon(f_\top(s))$ for $s \in S_{d(\top)}$ extending Def. 14 and Lem. 10 are defined in the obvious way, i.e. $f_\top(s) = \top$ for all $s \in S_{o(\top)}$ and $\sigma_{\top,(\alpha,*)}(\alpha\alpha', *) = (\alpha', *)$. We then use the following property of the maps $f_t$ in connection with substitution.

**Lemma 14.** *Let $t$ be a term of $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$. If $m \colon [\![t']\!] \to [\![t'']\!]$ is a morphism such that*

$$\forall s \in S_{d(t')}, \forall p \in Pos, \forall n > 1 \\ \big(\exists u''.f_{t''}(\mathcal{E}l(m)s)p = \epsilon_n u'' \Rightarrow \exists u'.f_{t'}(s)p = \epsilon_n u'\big)$$

*then*

$$\forall s \in S_{d(t[t'/x])}, \forall p \in Pos, \forall n > 1 \\ \big(\exists u''.f_{t[t''/x]}(\mathcal{E}l([\![t]\!]m)s)p = \epsilon_n u'' \Rightarrow \exists u'.f_{t[t'/x]}(s)p = \epsilon_n u'\big)$$

*Proof.* Assume that $m \colon [\![t']\!] \to [\![t'']\!]$ is a morphism such that

$$\forall s \in S_{d(t')}, \forall p \in Pos, \forall n > 1 \\ \big(\exists u''.f_{t''}(\mathcal{E}l(m)s)p = \epsilon_n u'' \Rightarrow \exists u'.f_{t'}(s)p = \epsilon_n u'\big)$$

By well founded induction we prove for $s \in S_{d(t[t'/x])}$ and $t \in \mathcal{T}_g^o$ with $FV(t) = \{x\}$ that

$$\forall p \in Pos, \forall n > 1\big(\exists u''.f_{t[t''/x]}(\mathcal{E}l(\llbracket t \rrbracket m)s)p = \epsilon_n u'' \Rightarrow \exists u'.f_{t[t'/x]}(s)p = \epsilon_n u'\big)$$

The well founded order is, as in Def. 14, given by $(s_1, t_1) < (s_2, t_2)$ if $h(s_1) < h(s_2)$ or $h(s_1) = h(s_2)$ and $gd(t_1) < gd(t_2)$.

We only use the lemma in two special cases, giving the two corollaries below.

**Corollary 3.** *Let $t$ be a term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$ and $m\colon \llbracket t' \rrbracket \to \llbracket \top \rrbracket$ the unique morphism into the terminal presheaf. Then*

$$\forall \phi \in Comp(\mathcal{O}_\epsilon(t[t'/x])),$$
$$f_{t[\top/x]_\infty}(\mathcal{E}l(\llbracket t \rrbracket m)_\infty \phi) \text{ is inadmissible } \Rightarrow f_{t[t'/x]_\infty}(\phi) \text{ is inadmissible.}$$

For $t$ a standard term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$ we define $t^0 = x$ and $t^{n+1} = t^n[t/x]$.

**Corollary 4.** *Let $t$ be a standard term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$ and let $\rho_t\colon \llbracket \mathsf{rec}\, x.t \rrbracket \to \llbracket t[\mathsf{rec}\, x.t/x] \rrbracket$ be the isomorphism given in Sec. 6.2. Then $\forall n \in \omega, \forall \phi \in Comp(\mathcal{O}_\epsilon(t^n[\mathsf{rec}\, x.t/x]))$,*

$$f_{t^{n+1}[\mathsf{rec}\, x.t/x]_\infty}(\mathcal{E}l(\llbracket t \rrbracket^n \rho_{t_1})_\infty \phi) \text{ is inadmissible } \Rightarrow f_{t^n[\mathsf{rec}\, x.t/x]_\infty}(\phi) \text{ is inadmissible.}$$

*Proof.* By definition $f_{\mathsf{rec}\, x.t}(s) = f_{t[\mathsf{rec}\, x.t/x]}\big(\mathcal{E}l(\rho_t)s\big)$ if $h(s) > 0$ and since $t$ is a standard term we have $\forall p \in Pos, f_{t[\mathsf{rec}\, x.t/x]}(\bot, *)p = \epsilon_n u \Rightarrow n = 0$, so we get that $\forall s \in S_{d(t[\mathsf{rec}\, x.t/x])} \forall p \in Pos \forall n > 1, f_{t[\mathsf{rec}\, x.t/x]}\big(\mathcal{E}l(\rho_t)s\big)p = \epsilon_n u \Rightarrow f_{\mathsf{rec}\, x.t}(s)p = \epsilon_n u$ and the desired result follows from Lem. 14 and Def. 9, by noting that $t^{n+1}[\mathsf{rec}\, x.t/x] = t^n[t[\mathsf{rec}\, x.t/x]/x]$ and $\llbracket t \rrbracket^n = \llbracket t^n \rrbracket$.

**Lemma 15.** *Let $t$ be a term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$. Then*

$$\forall t' \in \mathcal{T}_g \cup \{\top\}, Adm_{f\,d(t')} \subseteq Adm_{d(t')} \Rightarrow Adm_{f\,d(t[t'/x])} \subseteq Adm_{d(t[t'/x])}$$

*implies*

$$\forall t' \in \mathcal{T}_g \cup \{\top\}, \forall n \in \omega,$$
$$Adm_{f\,d(t')} \subseteq Adm_{d(t')} \Rightarrow Adm_{f\,d(t^n[t'/x])} \subseteq Adm_{d(t^n[t'/x])}$$

*Proof.* By an easy induction in $n$.

**Proposition 11.** *Let $t$ be a standard term in $\mathcal{T}_g^o$ such that $FV(t) \subseteq \{x\}$. Then for all $t'$ in $\mathcal{T}_g \cup \{\top\}$, $Adm_{f\,d(t')} \subseteq Adm_{d(t')}$ implies $Adm_{f\,d(t[t'/x])} \subseteq Adm_{d(t[t'/x])}$.*

*Proof.* (Sketch) By structural induction in $t$, using Lem. 4, Lem. 3 and Lem 15 above in the case for recursion.

If we take $t$ to be a closed term in the proposition above and e.g. $t' = \top$ then $t[t'/x]) = t$ so we get that $f_{t\infty}$ reflects admissibility, which was what we wanted to show.

**Corollary 5.** *Let t be a standard term in $\mathcal{T}_g$. Then $Adm_{f\,d(t)} \subseteq Adm_{d(t)}$.*