Towards Categorical Models for Fairness: Fully Abstract Presheaf Semantics of SCCS with Finite Delay ⋆

Thomas T. Hildebrandt [1]

Theory department, 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 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. This can be viewed as a first step towards representing *fairness* in categorical models for concurrency. The open map bisimulation is shown to coincide with *extended bisimulation* of Hennessy and Stirling, which is essentially fair CTL*-bisimulation. We give a *denotational* semantics of Milner's SCCS with finite delay in the presheaf model, which differs from previous semantics by giving the meanings of recursion by *final* coalgebras and meanings of finite delay by *initial* algebras of the process equations for delay. Finally we formulate Milner's 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 [1] 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 ϵt. The intended semantics is that ϵt can perform

⋆ Full and revised version of a paper appearing in Proceedings of CTCS'99.

[1] Supported by the RCES project funded by the Danish Research Councils.

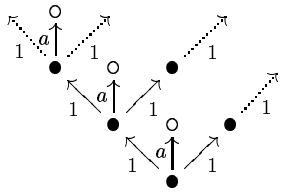
an unbounded number of 1-actions (delays) but *must eventually* perform an action, changing to an agent t', if t can perform this action and change to t', or it *must eventually stop*, if t cannot perform any actions. In other words, its actions are the same as for (the possibly infinite delay) $\delta t = \mathsf{rec}\, x.(1 : x + t)$, except that infinite unfolding of the recursion is not allowed.

To deal with agents in which only some infinite computations are admissible, one must re-address 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 [2], will identify agents that only differ on whether some infinite computations are admissible or not, in particular ϵt is identified with δt for any term t. Moreover, both ϵt and δt 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 [1] Milner proposes a behavioural 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 [3], 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 labeling 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^ω 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 [3] 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 [4] for *general transition systems*, which is essentially *fair CTL*-bisimulation* [5,6], except being formulated for *edge* labeled structures.

The present paper is a revised version of a paper appearing in Proceedings of CTCS'99 and the material also appears as a chapter in the authors PhD-thesis [7]. Its background is the work on presenting models for concurrency categorically as initiated by Winskel and Nielsen [8] and developed further in the work on bisimulation from open maps [9] and presheaf models for concurrency [10–13]. 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 identifications 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 [8] 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 [9]. Here one gets an abstract behavioural equivalence by choosing a *path category*, or to be a bit more general [14], a functor from a category of *path shapes* to the model at issue, identifying the *observable computations* (in [9] assumed to be the inclusion of a subcategory). The open maps approach gained ground through the further development [10,11,15,13,16] of the *presheaf* models for concurrency also proposed in [9]. Here one *starts* with a path category P (of non-empty path shapes) and then takes the category $\widehat{\mathsf{P}}$ of presheaves over P as model. The categorical justification [10,16] 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 initial object to 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 [11,13,10] 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) labeled 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 [9,16,11], the typical constructions of a CCS-like language can be expressed as functors preserving the canonical equivalence. As suggested in [9], 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* [17]. 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 [4], and it turns out that the extended bisimulation defined in [4] coincides with the abstract bisimulation obtained from open maps. We end by showing how to express Milner's [18] 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 [10] 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 ϵt 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 [3,19,20,4,21] 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 [3] and Hennessy and Stirling in [4]. The approach in [3] 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 initial algebras and final coalgebras. The semantics given in [19] is fully abstract with respect to the *fortification* equivalence, so it makes the non-intuitive identifi-

cations mentioned above. Moreover, it only covers *bounded non-determinism* as obtained from terms in which only a binary sum is allowed. The semantics of [20] is based on the fortification equivalence too. It is worth noting that for the models given in [19–21] 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 [1], the finite delay operator and how to derive a fair parallel [18]. 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 [4]. 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^ω refer to the sets of respectively all finite and all infinite sequences of actions. We let Fin and 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 α. For α, α' 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 α. If $\beta \in S^+$ and $\beta \leq \alpha$ in $S^+ \cup S^\omega$, we will write $\beta \leq_f \alpha$ for β is *finite and below* α.

1.1 Categories of Transition Systems and Synchronisation Trees

We here repeat the definition of transition systems given in [8] 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* $(S_T, i_T, \rightarrow_T$

$, Act)$, *where*

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

As usual we write $s \xrightarrow{a}_T s'$ *for* $\exists (s, a, s') \in \rightarrow_T$ *and let* $do\Big((s, a, s')\Big) = s$, $co\Big((s, a, s')\Big) = s'$, $act\Big((s, a, s')\Big) = a$. *The set of (finite or infinite)* computations *are given by*

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

and we let $Comp_{fin}(T) = Comp(T) \cap \rightarrow_T^*$. *The set of* runs *is given by*

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

and we let $Run_{fin}(T) = Run(T) \cap \rightarrow_T^*$ *and* $Run_{inf}(T) = Run(T) \cap \rightarrow_T^\omega$. *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 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 TS, with arrows being simulations.

Definition 2 *A simulation from a transition system* $T = (S_T, i_T, \rightarrow_T, Act)$ *to a transition system* $T' = (S_{T'}, i_{T'}, \rightarrow_{T'}, Act)$ *is a mapping* $\sigma \colon S_T \rightarrow S_{T'}$ *of states, such that*

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

We let ST refer to the full subcategory of TS with objects the synchronisation trees (with label set Act).

As shown in [8], the category ST is a coreflective subcategory of the category TS of transition systems; the inclusion $\mathsf{ST} \hookrightarrow \mathsf{TS}$ has a right adjoint $unf : \mathsf{TS} \rightarrow \mathsf{ST}$ which acts on objects by *unfolding* the transition system.

Definition 3 *Let* T *be a transition system with label set* Act. *Then the* unfolding *of* T *is the synchronisation tree*

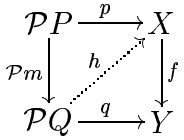
$$unf(T) = (Run_{fin}(T), \epsilon, \{\Big(\phi, a, \phi(s, a, s')\Big) \mid \phi(s, a, s') \in Run_{fin}(T)\}, Act),$$

where ϵ *is the empty transition sequence. Let* $m \colon T \rightarrow T'$ *be a morphism between two transition systems with underlying map* $\sigma \colon S_T \rightarrow S_{T'}$. *Then* $unf(m) \colon unf(T) \rightarrow unf(T')$ *is the synchronisation tree morphism with un-*

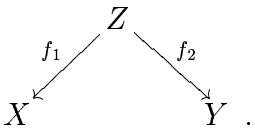
derlying map $\sigma_*\colon Run_{fin}(T) \to Run_{fin}(T')$ *given by* $\sigma_*(\phi) = \phi'$, *such that* $|\phi| = |\phi'|$ *and for all* $i < |\phi|$, *if* $\phi_i = (s, a, s')$ *then* $\phi'_i = \big(\sigma(s), a, \sigma(s')\big)$.

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 [9]. Given a model 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 M, identifying the *observable computations* (in [9] assumed to be the inclusion of a subcategory). A map $f\colon X \to Y$ in M is then defined to be $\mathcal{P}$*-open* (P-open or just open if the embedding is clear from the context) if whenever for a morphism $m\colon P \to Q$ in P and morphisms $p\colon \mathcal{P}P \to X$ and $q\colon \mathcal{P}Q \to Y$ such that the diagram



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 are then defined to be $\mathcal{P}$-bisimilar if they are related by a *span* of $\mathcal{P}$-open maps

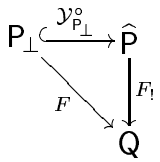


.

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, it was shown in [9] that $\mathcal{B}ran$-bisimulation coincides with Park and Milner's strong bisimulation on labeled transition systems [22,1]. In subsequent work [23], 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 [9], presheaf categories were suggested as abstract models for concurrency, equipped with a *canonical* notion of bisimulation equivalence. For P a small category, the category $\widehat{\mathsf{P}}$ of *presheaves over* P has as objects all functors $X\colon \mathsf{P}^{\mathsf{op}} \to \mathsf{Set}$ (where Set is the category of sets and functions) and as arrows natural transformations between such. Let us briefly repeat from [10,16] the categorical justification for taking presheaf categories as models for concurrency. We write $\mathsf{P}_\perp$ for the category obtained by adding a (new) initial object $\perp$ to P and $\mathcal{Y}^\circ_{\mathsf{P}_\perp}\colon \mathsf{P}_\perp \hookrightarrow \widehat{\mathsf{P}}$ for the strict extension of the well known

Yoneda embedding $\mathcal{Y}_{\mathsf{P}}\colon \mathsf{P} \hookrightarrow \widehat{\mathsf{P}}$, taking $\bot$ to the initial, i.e. constant empty, presheaf. Then $\mathcal{Y}^{\circ}_{\mathsf{P}_\bot} : \mathsf{P}_\bot \hookrightarrow \widehat{\mathsf{P}}$ is the *free connected-colimit completion* of $\mathsf{P}_\bot$; it is the category obtained (up to equivalence) by freely adding all colimits of connected diagrams to $\mathsf{P}_\bot$ and any functor $F\colon \mathsf{P} \to \mathsf{Q}$ for Q a category having all connected colimits, can be extended freely (as a left Kan extension [24]) to a connected-colimit preserving functor $F_!\colon \widehat{\mathsf{P}} \to \mathsf{Q}$ making the diagram



(3)

commute. The embedding $\mathcal{Y}^{\circ}_{\mathsf{P}_\bot} : \mathsf{P}_\bot \hookrightarrow \widehat{\mathsf{P}}$ provides a *canonical* choice of path shapes for a presheaf model $\widehat{\mathsf{P}}$, and thus a canonical notion of bisimulation. As recalled below, the category $\widehat{\mathsf{Fin}}$ is indeed equivalent to the category of synchronisation trees given above, and the canonical bisimulation is the usual strong bisimulation. By replacing sequences with partial orders one obtain models and bisimulations generalising event-structures and history-preserving bisimulation, see e.g. [9].

Remark 4 *In [9], focus was put on path categories with an initial object $\bot$ and* rooted *presheaves, that is, presheaves for which $X(\bot)$ is the singleton set. The canonical bisimulation was then taken to be that given by span of* surjective $\mathcal{Y}_{\mathsf{P}_\bot}$-open maps. *It is easy to verify that the category of rooted presheaves over $\mathsf{P}_\bot$ is equivalent to the category $\widehat{\mathsf{P}}$. Moreover, surjective $\mathcal{Y}_{\mathsf{P}_\bot}$-open maps between rooted presheaves in $\widehat{\mathsf{P}_\bot}$ corresponds via the equivalence exactly to $\mathcal{Y}^{\circ}_{\mathsf{P}_\bot}$-open maps in $\widehat{\mathsf{P}}$.*

The following proposition [25,13,10] is one of the most important results about open map bisimulation in presheaf models.

Proposition 5 *Let $\mathcal{F}\colon \widehat{\mathsf{P}} \to \widehat{\mathsf{Q}}$ be a connected-colimit preserving functor and $m\colon X \to Y$ a $\mathcal{Y}^{\circ}_{\mathsf{P}_\bot}$-open map in $\widehat{\mathsf{P}}$. Then $\mathcal{F}m\colon \mathcal{F}X \to \mathcal{F}Y$ is a $\mathcal{Y}^{\circ}_{\mathsf{Q}_\bot}$-open map in $\widehat{\mathsf{Q}}$.*

It follows that connected-colimit preserving functors between presheaf categories preserve the canonical notion of bisimulation. To appreciate this result, note that it has been shown that most of the typical constructions of CCS-like and Dataflow languages can be expressed as (component-wise) connected-colimit preserving functors [11,12,16], for which it follows directly from the above result that the canonical bisimulation is a congruence.

Below we repeat briefly the concrete representation of presheaves as transition systems as given in [26].

Notation 6 *For $q \leq p$ in a partial order P, let $[q,p]\colon q \to p$ denote the unique arrow from q to p in P viewed as a category and $[p,q]\colon p \to q$ the unique arrow from p to q in P^{op}. For a partial order P, a presheaf $X\colon \mathsf{P}^{\mathsf{op}} \to \mathsf{Set}$, objects $q \leq p$ in P and an element $x \in X(p)$ we will employ the standard notation [17], writing $x \cdot [q,p]$ for the element $X([p,q])x$. This element is referred to as* the restriction of x to q.

For a presheaf X in $\widehat{\mathsf{Fin}}$, its corresponding synchronisation tree is constructed from *the category of elements* [17]. The set of states is given by the *elements* of X with an initial state added and the transition relation is defined from the restriction action of the presheaf.

Definition 7 *For a presheaf X in $\widehat{\mathsf{Fin}}$, the* synchronisation tree corresponding to X *is given by $\mathcal{E}l(X) = (S, i, \to, Act)$, where*

$$S = \Big\{(\alpha, x) \mid \alpha \in \mathsf{Fin} \text{ and } x \in X(\alpha)\Big\} \cup \{i\}, \quad \text{and}$$
$$\to = \Big\{((\alpha, x), a, (\alpha a, x')) \mid x' \cdot [\alpha, \alpha a] = x\Big\} \cup \{(i, a, (a, x)) \mid a \in Act \wedge x \in X(a)\}.$$

1.3 Initial Algebras and Final Coalgebras

Below we recall the categorical analogues of pre- and post-fixed points [27].

Definition 8 *Let $F\colon \mathsf{P} \to \mathsf{P}$ be an endofunctor. A* co-algebra *for F is a pair (p, m) of an object and a morphism of P such that $m\colon p \to F(p)$. Dually, an* algebra *for F is a pair (p, m) such that $m\colon F(p) \to p$. The co-algebras of F form the objects of a category $\mathsf{F}_{\mathsf{coAlg}}$, with arrows $f\colon (p, m) \to (q, n)$ being arrows $f : p \to q$ of P such that*

$$\begin{array}{ccc} p & \xrightarrow{m} & F(p) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle F(f)} \\ q & \xrightarrow{n} & F(q) \end{array}$$

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 9 (Lambek) *Let $F\colon \mathsf{P} \to \mathsf{P}$ be an endofunctor. 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. If (q, n) is another initial algebra for F, then q is isomorphic to p. The dual statement holds for final co-algebras. If F has an initial algebra, let μF denote the (unique up to isomorphism) initial algebra. Similarly, let νF 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 10 *Let* P *be a category with terminal object* $\top$ *and* $F\colon \mathsf{P} \to \mathsf{P}$ *an endofunctor on* P*. If the* ω^{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 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. [27].

Lemma 11 *Let* P *be a category with initial object* $\bot$ *and* $F\colon \mathsf{P} \to \mathsf{P}$ *an endofunctor on* P*. If the* ω*-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 morphism* $m\colon F(P) \to P$ *is an initial algebra.*

2 Synchronous CCS with Finite Delay

In this section we recall Milner's calculus SCCS [1] of *synchronous* CCS and the finite delay operator [18] from which one can encode a CCS-like calculus with a fair asynchronous parallel composition. Assume a distinguished element $1 \in Act$ such that $(Act, \bullet, 1)$ is an Abelian[2] monoid with 1 being the identity. The *basic* operators of SCCS are action prefixing, synchronous product, nondeterministic choice and restriction. Formally, the terms are given by

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

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.

[2] i.e. commutative

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

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

Fig. 1. Operational semantics of SCCS

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. 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.

Definition 12 *Let t be a term in $\mathcal{T}$. Then the* derivation transition system *for t is the (reachable) transition system $D(t) = (S, t, \rightarrow_t, Act)$, where $S = \left\{t' \in \mathcal{T} \mid t \rightarrow^* t'\right\}$, i.e. all states reachable from t by the relation $\rightarrow$ defined by the rules in Fig. 1 and $\rightarrow_t = \rightarrow \cap S \times Act \times S$.*

In [1] a *delay* operator δ was introduced in order to encode a CCS-like calculus with the usual asynchronous parallel composition. For a process t, define $\delta t = \mathsf{rec}\, x.(1{:}x{+}t)$. In the standard semantics, δt is the (unique up to bisimulation) fixed point of the process equation

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

By defining asynchronous prefixing by

$$a.t = a{:}\delta t \ , \tag{5}$$

one can encode an asynchronous parallel composition by

$$t \mid\mid t' = (\delta t \times t') + (t \times \delta t') \ . \tag{6}$$

As an economical way to be able to express that some infinite runs are inadmissible, Milner introduces in [18] a *finite, but unbounded delay* operator ϵ (expectation). The idea was, that the finite delay could be taken as the *only* operator giving rise to inadmissible infinite runs, in particular recursion gives rise to admissible infinite runs. The immediate actions are the same for finite

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

Fig. 2. Derivation Rules for Finite Delay

delay as for the derived delay operator, which can be described by the rules given in Fig. 2. However, *infinite* waiting is ruled out as inadmissible. In other words, fulfillment of the delay is always expected. We will formalize this in Sec. 5 below. By replacing the derived delay operator by the finite delay operator in (5) and (6) above one now gets a *fair* asynchronous prefix and parallel composition.

We will let SCCSϵ and $\mathcal{T}_\epsilon$ refer to respectively the calculus SCCS extended with the finite delay operator ϵ and the set of closed 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 *in-admissibility* of infinite computations can be expressed.

3 Observing Infinite Computations

We approach a categorical model for infinite computations by studying presheaves, and sheaves, over the path category Inf obtained by adding infinite paths to the path category Fin. This fits with the spirit of [4], 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

A presheaf $X : \mathsf{Inf}^{op} \to \mathsf{Set}$ in $\widehat{\mathsf{Inf}}$ restricts to a presheaf in $\widehat{\mathsf{Fin}}$ by composing it with the inclusion of Fin into Inf. Now, for $\alpha \in Act^\omega$, an element $x \in X(\alpha)$ will specify a unique infinite path in the synchronisation tree corresponding to the restriction of X to $\widehat{\mathsf{Fin}}$. 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 sequences. We will 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. We therefore take the subcategory of presheaves with at most 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 Inf with respect to a simple Grothendieck topology for 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) *amalgamations*).

Definition 13 *Let* $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ *denote the* separated presheaves*, which is the full subcategory of* $\widehat{\mathsf{Inf}}$ *with objects the presheaves* X *satisfying that for all* $\alpha \in Act^{\omega}$ *and elements* $x, x' \in X(\alpha)$*,*

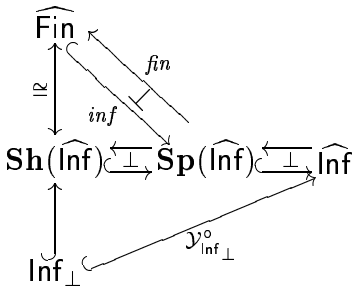
- $(\forall \beta \leq_f \alpha.x \cdot [\beta, \alpha] = x' \cdot [\beta, \alpha]) \implies x = x'$ *(Separated)*

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

Sheaves, separated presheaves and presheaves are known to be closely related and rich in structure [17,28]. 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 particularly 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 infinite sequences.

We also have that the objects of $\mathsf{Inf}_{\perp}$ under the strict extension of the Yoneda embedding are sheaves.[3] Together with Prop. 14, 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.



(7)

[3] In fact the Grothendieck topology we use is the *canonical* topology for Inf [17], 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 $\mathit{fin} \dashv \mathit{inf}$ refer to the reflection between $\widehat{\mathsf{Fin}}$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ obtained via the equivalence between $\mathbf{Sh}(\widehat{\mathsf{Inf}})$ and $\widehat{\mathsf{Fin}}$.

For more details on Grothendieck topologies, sheaves and separated presheaves see [17]. 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 $C^\bullet \subseteq Comp(T)$ is the least set including C such that

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

The two first conditions ensure that the definition fits with that of *general transition systems* in [4]. The last condition restricts attention to the special case where any finite computation is admissible.

Definition 15 *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* $Adm_G = Comp(T)$*. We write* $\mathit{fin}(G) = (S_G, i_G, \rightarrow_G, Act)$ *for the underlying transition system of* G.

Generalised transition systems (with label set Act) forms the objects of a category GTS with morphisms defined as follows.

Definition 16 *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 σ_∞ *is the extension of* σ *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* $\phi'_i = \Big(\sigma(s), a, \sigma(s')\Big)$.

We let GST refer to the full subcategory of GST with objects the generalised synchronisation trees (with label set Act).

The following lemma gives some alternative definitions of morphisms between generalised transition systems. In particular it shows that the morphisms of 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 17 *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 TS into GTS that maps a plain transition system to the corresponding standard generalised transition system.

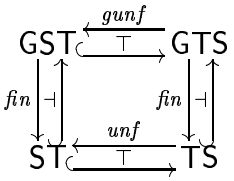
Proposition 18 *The functor* $fin\colon \mathsf{GTS} \to \mathsf{TS}$ *is a left adjoint to the inclusion* $inf\colon \mathsf{TS} \hookrightarrow \mathsf{GTS}$ *which maps a transition system* $T = (S_T, i_T, \to_T, Act)$ *to the standard generalised transition system* $(S_T, i_T, \to_T, Comp(T), Act)$ *and maps a transition system morphism to the generalised transition system morphism with the same underlying map of states.*

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

Proposition 19 *The inclusion functor* $\mathsf{GST} \hookrightarrow \mathsf{GTS}$ *has a right adjoint* $gunf\colon \mathsf{GTS} \to \mathsf{GST}$ *such that the diagram*

$$\begin{array}{ccc} \mathsf{GST} & \xleftarrow{gunf} & \mathsf{GTS} \\ {\scriptstyle fin}\downarrow & & \downarrow{\scriptstyle fin} \\ \mathsf{ST} & \xleftarrow{unf} & \mathsf{TS} \end{array}$$

commutes, where unf is the unfolding of transition systems defined in Sec. 1. In fact all four squares in the diagram



commute.

We will now give a concrete representation of presheaves in $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ as generalised synchronisation trees.

To generalise the construction of a synchronisation tree from a presheaf we gave in Sec. 1 we use the following property stated in the lemma below: In a reachable generalised transition system, the set of admissible computations is determined uniquely by the set of admissible *infinite runs*.

Lemma 20 *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 $C = A^\bullet$.*

We can now define 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.

Definition 21 *For a presheaf X in $\widehat{\mathsf{Inf}}$, the* generalised synchronisation tree corresponding to X *is given by*

$$\mathcal{E}l(X) = (S, i, \to, Act, Adm),$$

where

$$S = \Big\{(\alpha, x) \mid \alpha \in \mathsf{Fin} \text{ and } x \in X(\alpha)\Big\} \cup \{i\},$$

$$\to = \Big\{((\alpha, x), a, (\alpha a, x')) \mid x' \cdot [\alpha, \alpha a] = x\Big\} \cup \{(i, a, (a, x)) \mid a \in Act \wedge x \in X(a)\},$$

and

$$Adm = \Big\{\phi \in Comp(T) \mid \exists \alpha \in Act^\omega \exists x \in X(\alpha).\ x \text{ is a limit point of } \phi\Big\}^\bullet,$$

where "x is a limit point of ϕ", means that $\forall j \in \omega. co(\phi_j) = \Big(\alpha(j), x \cdot [\alpha(j), \alpha]\Big)$.

From the embedding $\mathcal{GB}ran_+\colon \mathsf{Inf} \hookrightarrow \mathsf{GTS}$ which maps a non-empty, finite or infinite sequence α to the standard generalised transition system (in fact generalised synchronisation tree) with exactly the one branch corresponding to α, we get a canonical functor [9] from GTS to $\widehat{\mathsf{Inf}}$, which 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 22 *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 $\mathbf{Sp}(\widehat{\mathsf{Inf}})$.*

The canonical functor gives the other direction of the equivalence forming the concrete representation of $\mathbf{Sp}(\widehat{\mathsf{Inf}})$.

Theorem 23 *The categories GST and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ are equivalent.*

In the light of Thm. 23 above, the functors $fin \colon \mathsf{GST} \to \mathsf{ST}$ and $inf \colon \mathsf{ST} \to \mathsf{GST}$ are just concrete representations of the reflection between $\widehat{\mathsf{Fin}}$ and $\mathbf{Sp}(\widehat{\mathsf{Inf}})$ given in (7).

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 (7) 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 separated presheaves. The category $\mathsf{Inf}_\perp$ embeds into the category of generalised transition systems by the functor $\mathcal{GB}ran \colon \mathsf{Inf}_\perp \hookrightarrow \mathsf{GTS}$, which is simply the strict extension of $\mathcal{GB}ran_+ \colon \mathsf{Inf} \hookrightarrow \mathsf{GTS}$ 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 [4].

Remark 24 *Extended bisimulation is essentially* fair CTL*-bisimulation *[29,5], except for being formulated for* edge *labeled 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 [6].*

We first give a characterisation of the $\mathsf{Inf}_\perp$-open maps of GTS that generalises the "zig-zag" morphisms in [9].

Proposition 25 *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 $\sigma \colon T \to U$. Then σ 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 σ 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 $\exists \phi \in Adm_T$ *such that* $\phi = s \stackrel{a_1}{\rightarrow}_T s_1 \stackrel{a_2}{\rightarrow}_T s_2 \stackrel{a_3}{\rightarrow}_T \ldots \stackrel{a_n}{\rightarrow}_T s_n \stackrel{a_{n+1}}{\rightarrow}_T \ldots$ *and for all* $j \in \omega$, $\sigma(s_j) = s'_j$.

Now we give the definition of extended bisimulation from [4] reformulated as a relation between two generalised transition systems (exploiting condition C3).

Definition 26 *([4]) 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* $\phi \in Adm_T$ *such that* $\phi_0 = s$, *then there exists a computation* $\phi' \in Adm_{T'}$ *such that* $|\phi| = |\phi'|$ *and* $\phi'_0 = s'$ *and for* $0 \leq j < |\phi|$, $act(\phi_j) = act(\phi'_j)$ *and* $(\phi_j, \phi'_j) \in R$,

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

Note that (by condition C3) extended bisimulation specialises to the standard strong 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. 25 it follows that extended bisimulation coincides with $\mathsf{Inf}_\perp$-bisimulation for generalised transition systems.

Proposition 27 *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 28 *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 [9] 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 [18] in terms of generalised transition systems.

The inadmissible infinite computations are identified in [18] via the notions

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

Fig. 3. Derivation rules for annotated finite delay

of *waiting computations*, *subagents* and *subcomputations*, which we will recall briefly. A computation $t_0 \to t_1 \to t_2 \to \ldots$ of an agent t_0 is said to be 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 ϵt have only themselves as subagent, $t \lfloor A$ 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 be able to record if the (Wait) rule was used to infer an action of a subagent, we annotate terms of the form ϵt with a subscript number $n \in \omega$, written $\epsilon_n t$, and replace the derivation rules of Fig. 2 by the rules in Fig. 3. The intuition is that the number records for how long the agent has been delaying. We will say that an agent t is *waiting* if $t = \epsilon_n t'$ for some term t' and $n > 1$.

In the following we let $\mathcal{T}_\epsilon$ refer to the set of annotated closed terms of SCCSϵ and we let $\epsilon_0 t$ and ϵt refer to the same agent. Note that any function with domain $\mathcal{T}$ can be regarded as a function with domain $\mathcal{T}_\epsilon$ by discarding the annotations.

We formalise the notion of subagent at a specific position as follows.

Definition 29 *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*

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

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

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

Definition 30 *An infinite computation $t_0 \xrightarrow{\alpha_0} t_1 \xrightarrow{\alpha_1} t_2 \xrightarrow{\alpha_2} \ldots$ 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 [18] 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. 30 above.

Definition 31 *Let* t *be a term in* $\mathcal{T}_\epsilon$. *Then the* derivation transition system *for* t *is the reachable* generalised *transition system*

$$\mathcal{O}_\epsilon(t) = (S, t, \rightarrow_t, Adm, Act) \ .$$

The set of states is the states reachable by the transition relation defined by the rules in Fig. 1 and Fig. 3

$$S = \{t' \mid t \rightarrow^* t'\} \ ,$$

the transition relation is the restriction of $\rightarrow$ *to states in* S

$$\rightarrow_t = \rightarrow \cap S \times Act \times S \ ,$$

and the set of admissible computations

$$Adm \subseteq Comp\Big((S, t, \rightarrow_t, Act)\Big)$$

is the set defined in Def. 30.

Remark 32 *We do not need to record* exactly *how many steps an agent has waited, just if it 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* $\epsilon_n t \xrightarrow{1} \epsilon_{\min\{n+1,2\}} t$ *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. This is a key step toward proving decidability of the extended bisimulation for a non trivial subset of SCCS*ϵ.

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 SCCSϵ.

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 Kan extension $(-)_!$ described in Sec. 1.

For $A \subseteq Act$, the *restriction on sequences* $(-)\restriction A \colon \mathsf{Inf} \to \mathsf{Inf}_\perp$ maps a sequence α to the (possible empty) longest prefix of α belonging to A^*. Formally, define $\alpha \restriction A = \alpha'$, where $\alpha' \in A^*$ is the unique sequence $\alpha' \leq \alpha$ such that $(\alpha = \alpha' a \alpha'' \implies a \notin A)$.

For $a \in Act$, the *action prefix on sequences* $a \colon \mathsf{Inf}_\perp \to \mathsf{Inf}$ maps a (possibly empty) sequence α 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. Formally, for $\alpha, \beta \in \mathsf{Inf}$, define $\alpha \bullet \beta = \gamma$, where γ 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 the partial orders viewed as categories. By (implicitly) composing with the embeddings $\mathcal{Y}^\circ_{\mathsf{Inf}_\perp} : \mathsf{Inf}_\perp \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\mathcal{Y}_{\mathsf{Inf}} \colon \mathsf{Inf} \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}})$, we get functors $(-)\restriction A \colon \mathsf{Inf} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$, $a \colon \mathsf{Inf}_\perp \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\bullet \colon \mathsf{Inf} \times \mathsf{Inf} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$. Applying the proper Kan extensions we get functors $(-)\restriction A_! \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$, $a_! \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ and $\bullet_! \colon \widehat{\mathsf{Inf} \times \mathsf{Inf}} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$. Finally, we precompose $\bullet_! \colon \widehat{\mathsf{Inf} \times \mathsf{Inf}} \to \mathbf{Sp}(\widehat{\mathsf{Inf}})$ with the (component-wise connected colimit-preserving [11]) functor $w \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \times \mathbf{Sp}(\widehat{\mathsf{Inf}}) \to \widehat{\mathsf{Inf} \times \mathsf{Inf}}$ defined (on objects) by

$$w(X, Y)(\alpha, \beta) = X(\alpha) \times Y(\beta) \ .$$

This gives us the following denotations of basic operators.

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

$$\mathcal{I}[\![\Sigma_{i \in I} t_i]\!] = \Sigma_{i \in I} \mathcal{I}[\![t_i]\!], \tag{8}$$
$$\mathcal{I}[\![a{:}t]\!] = a_! \mathcal{I}[\![t]\!], \tag{9}$$
$$\mathcal{I}[\![t \times t']\!] = \bullet_! \circ w(\mathcal{I}[\![t]\!], \mathcal{I}[\![t']\!]), \tag{10}$$
$$\mathcal{I}[\![t \restriction A]\!] = \mathcal{I}[\![t]\!] \restriction A_!, \tag{11}$$

The semantic functions are extended in the standard way to terms with free variables in a set $\mathcal{V}$, yielding functors

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

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

The first three definitions (8)-(10) 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 sum and $\times$ are used to indicate clearly how an element came about, which we will use in App. B.

$$[\![t\lfloor A]\!]\alpha = \{e \mid \alpha \in A^\infty \text{ and } e \in [\![t]\!]\alpha\}. \tag{12}$$

$$[\![\Sigma_{i\in I} t_i]\!]\alpha = \Big\{(\mathsf{sum}\ i, (\alpha, e)) \mid i \in I \text{ and } e \in [\![t_i]\!]\alpha\Big\}. \tag{13}$$

$$[\![a\!:\!t]\!]\alpha = \begin{cases} \lfloor [\![t]\!] \rfloor \alpha' & \text{if } \alpha = a\alpha', \\ \emptyset & \text{otherwise,} \end{cases} \tag{14}$$

where $\lfloor - \rfloor \colon \mathbf{Sp}(\widehat{\mathsf{Inf}}) \hookrightarrow \mathbf{Sp}(\widehat{\mathsf{Inf}_\perp})$ is the obvious "lifting" functor, which can be represented explicitly by

$$\lfloor X \rfloor \alpha = \begin{cases} \{*\} & \text{if } \alpha = \perp, \\ X\alpha & \text{otherwise.} \end{cases} \tag{15}$$

$$\begin{aligned} [\![t_1 \times t_2]\!]\alpha = \{&(\beta, e_1) \times (\gamma, e_2) \mid \\ &\beta, \gamma \in \mathsf{Inf}. \beta \bullet \gamma = \alpha \text{ and } e_1 \in [\![t_1]\!]\beta \text{ and } e_2 \in [\![t_2]\!]\gamma\}. \end{aligned} \tag{16}$$

6.2 Semantics of Recursion

For *recursion* we need to take care. Taking least fixed points, i.e. initial algebras, as the meanings of recursion would not reflect that it is indeed admissible to unfold a recursion infinitely. An example that illustrates this is given below, where we see 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. 10 given in Sec. 1 to construct final co-algebras for all relevant endofunctors as limits of ω^{op}-chains. The definition is then extended to processes with more than one variable in the usual way as a

limit with parameters [24]. From the explicit definitions given in Eq. (12)-(16) we can show that all basic operators preserve ω^{op}-limits. From the general fact that limits commute with limits [24] we get that recursion preserves ω^{op}-limits as well, i.e. if $\mathsf{rec}\, x.t$ has free variables then $\mathcal{I}[\![\mathsf{rec}\, x.t]\!]$ preserves ω^{op}-limits.

Lemma 33 *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 ω^{op}-limits then

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

(is well defined and) preserves ω^{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 defining $\top\alpha = \{*\}$. Now we use the explicit definition of limits in the category Set to define

$$[\![\mathsf{rec}\, x.t]\!]\alpha = \Big\{\langle e_0, e_1, \ldots, e_n, \ldots\rangle \in \prod_{n \in \omega} [\![t]\!]^n(\top)\alpha \mid [\![t]\!]^n(\tau)_\alpha e_{n+1} = e_n\Big\}, \quad (17)$$

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

$$\begin{array}{ccc} [\![\mathsf{rec}\, x.t]\!] & \xrightarrow{\pi_{n+1}} & [\![t]\!]^{n+1}(\top) \\ {\scriptstyle \rho_t}\downarrow & \nearrow_{[\![t]\!](\pi_n)} & \\ [\![t]\!]([\![\mathsf{rec}\, x.t]\!]) & & \end{array} \quad (18)$$

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

We have now given semantics to all operators in SCCSϵ except for finite delay. 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 term $\mathsf{rec}\, x.x$, which according to the operational semantics denotes the process that cannot perform any actions, which is the *initial* object in $\widehat{\mathsf{Inf}}$. It is not difficult to compute the appropriate limit finding that $\mathcal{I}[\![\mathsf{rec}\, x.x]\!] \cong \top$, that is, the *final* object in $\widehat{\mathsf{Inf}}$. However, if we restrict the language to only allow *guarded* recursion, we can prove the desired correspondence.

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. 11 since the denotation of prefixing preserves connected colimits and the denotation of sum all colimits. The definition is extended to open terms (in which x is not free) as a colimit with parameters.

From the explicit definition of colimits in Set, we find that we can take

$$[\![\epsilon t]\!]\alpha = \left\{ \Big(\mathsf{del}\ n, (\alpha', e) \Big) \mid n \in \omega,\ \alpha = 1^n \alpha' \text{ and } e \in \lfloor\!\lfloor [\![t]\!] \rfloor\!\rfloor \alpha' \right\} \tag{19}$$

as explicit definition of finite delay on objects (again the tag 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 $e \in \lfloor\!\lfloor [\![t]\!] \rfloor\!\rfloor \alpha'$.

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

Lemma 34 *Let t be a (possibly open) term of SCCSϵ 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 ω^{op}-limits then

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

(is well defined and) preserves ω^{op}-limits.

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

From the fact that the denotations (in $\widehat{\mathsf{Inf}}$) of all basic operators are built from (component-wise) 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 [9] we get that they preserve open maps 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 (alternatively one could use the same technique as used in [11] for showing that denotations of recursions (given by initial algebras) preserve open maps). This gives us the following result.

Proposition 35 *Extended bisimulation is a congruence with respect to all basic operators of SCCSϵ as well as finite delay.*

When it comes to *recursion* we meet a problem, namely to identify the "right" notion of bisimulation (from open maps) for denotations of open terms, that is, functors between presheaf categories. In [11], the notion of open maps is extended to natural transformations, which is said to be open if all components are open maps. This is shown to be sufficient to guarantee that open map bisimulation is a congruence with respect to the denotations of recursion (given by *initial* algebras) in a CCS-like calculus, but it seems not to be sufficient to give the desired congruence property for recursion given by final co-algebras.

In [13,10] it is observed that connected colimit preserving functors between presheaf categories can be regarded as objects of a presheaf category and thus comes with a canonical notion of open maps. This gives a slightly stronger notion of open maps. However, we have not been able to show that the denotations yield connected colimit preserving functors.

Consequently, the question of finding a notion of bisimulation for the denotations of open terms, which is a congruence with respect to recursion, remains an unsolved question.

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 GTS. First of all we will restrict attention to terms with only *guarded recursion*, for the reason given in Sec. 6.2 above. Recall from e.g. [1] that a recursion $\mathsf{rec}\,x.t$ is guarded, if all free occurrences 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 SCCSϵ 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 *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.

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 36 *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. 27 and Cor. 28 in Sec. 4 we can now deduce the desired result.

Theorem 37 *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 [8–10], 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 [4] 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 [18] 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 [3], more intuitive, equivalence than the one obtained from the fortification preorder in [18], which except for [3] has been the basis for previous denotational semantics of SCCS with finite delay [20,19,30]. Benefitting from the categorical presentation, our semantics appears to give a conceptually simpler treatment of infinite computations than the one in [3].

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 [30]. A notion of open maps between denotations of open terms stronger than the one in [11] is currently being explored, which hopefully is a congruence with respect to recursion. We also hope to be able to extend the presheaf model for (finitary) dataflow given in [12] to infinite computations along the lines of the present paper, giving a model of dataflow in which fairness, and in particular *fair merge* [31], can be expressed. We get a characteristic HML-like path logic [9] for extended bisimulation from the open maps approach. This logic should be compared to the characteristic logic given in [4] and an edge-labeled version of the (fair) CTL* logic. Being essentially CTL*-bisimulation, extended bisimulation is decidable for SCCSϵ processes giving rise to finite generalised transition systems, if the characterisation of admissible infinite computations can be described as a Muller condition. Finally, 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. [32].

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

References

[1] R. Milner, Calculi for synchrony and asynchrony, Theoretical Computer Science 25 (1983) 267–310.

[2] R. Milner, A calculus of communicating systems (1980).

[3] P. Aczel, A semantic universe for fairness, preliminary Draft (1996).

[4] M. Hennessy, C. Stirling, The power of the future perfect in program logics, Information and Control (1985) 23–52.

[5] E. A. Emerson, Temporal and modal logic, in: J. van Leeuven (Ed.), Handbook of Theoretical Computer Science, Vol. 2, Elsevier Science B.V., 1990, Ch. 16, pp. 996–1072.

[6] A. Aziz, V. Singhal, F. Balarin, Equivalences for fair kripke structures, in: S. Abiteboul, E. Shamir (Eds.), ICALP '94: International Colloquium on Automata, Languages, and Programming, Vol. 820 of LNCS, Springer-Verlag, 1994, pp. 364–375.

[7] T. T. Hildebrandt, Categorical models for concurrency: Independence, fairness and dataflow, Ph.D. thesis, Department of Computer Scence, University of Aarhus, Denmark (October 1999).

[8] G. Winskel, M. Nielsen, Handbook of Logic in Computer Science, Vol. IV, OUP, 1995, Ch. Models for concurrency, pp. 1–148.

[9] A. Joyal, M. Nielsen, G. Winskel, Bisimulation from open maps, LICS '93 special issue of Information and Computation 127 (2) (1996) 164–185, available as BRICS report, RS-94-7.

[10] G. L. Cattani, Presheaf models for concurrency, Ph.D. thesis, Aarhus University (1999).

[11] G. L. Cattani, G. Winskel, Presheaf models for concurrency, in: CSL'96, Vol. 1258 of LNCS, Springer, 1997, pp. 58–75.

[12] T. T. Hildebrandt, P. Panangaden, G. Winskel, A relational model of non-deterministic dataflow, in: CONCUR'98, Vol. 1466 of LNCS, Springer-Verlag, 1998, pp. 613–628.

[13] G. Winskel, A linear metalanguage for concurrency, in: AMAST '98, Vol. 1548 of LNCS, Springer-Verlag, 1998, pp. 42–58.

[14] M. Fiore, G. L. Cattani, G. Winskel, Weak bisimulation and open maps, in: Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science, 1999, pp. 67–76.

[15] G. L. Cattani, I. Stark, G. Winskel, Presheaf models for the pi-calculu, in: CTCS'97, Vol. 1290 of LNCS, Springer, 1997, pp. 106–126.

[16] G. L. Cattani, G. Winskel, Presheaf models for CCS-like languages, Submitted .

[17] S. Mac Lane, I. Moerdijk, Sheaves in Geometry and Logic: A First Introduction to Topos Theory, Springer, 1992.

[18] R. Milner, A finite delay operator in synchronous CCS, Tech. Rep. 116-82, University of Edinburgh, Deptartment of Computer Science, Kings Buildings (1982).

[19] M. Huth, M. Kwiatkowska, Finite but unbounded delay in synchronous CCS, in: A. Edalat, S. Jourdan, G. McCusker (Eds.), Advanced methods in theory and formal methods of computing: Proceedings of the third Imperial College workshop April 1996, Imperial College Press, 1996, pp. 312–323.

[20] M. Hennessy, Modelling finite delay operators, Tech. rep., University of Edinburgh (1983).

[21] G. Winskel, Generalised synchronisation trees, handwritten notes (1983).

[22] D. Park, Concurrency and automata on infinite sequences, in: Theoretical Computer Science: 5th GI-Conference, Vol. 104 of LNCS, Springer, 1981, pp. 168–183.

[23] A. Cheng, M. Nielsen, Open maps (at) work, Research Series RS-95-23, BRICS, Department of Computer Science, University of Aarhus (Apr. 1995).

[24] S. Mac Lane, Categories for the Working Mathematician, Springer, 1971.

[25] G. L. Cattani, A. J. Power, G. Winskel, A categorical axiomatics for bisimulation, in: Proceedings of the 9th International Conference on Concurrency Theory, CONCUR '98, Vol. 1466 of LNCS, Springer-Verlag, 1998, pp. 581–596.

[26] G. Winskel, M. Nielsen, Presheaves as transition systems, in: D. Peled, V. Pratt, G. Holzmann (Eds.), POMIV'96, Vol. 29 of DIMACS, AMS, 1996, pp. 129–140.

[27] Barr, Wells, Category Theory for Computing Science, Prentice Hall, 1990.

[28] O. Wyler, Lecture Notes on Topoi and Quasitopoi, World Scientific, 1991.

[29] E. A. Emerson, J. Y. Halpern, "sometimes" and "not never" revisited: On branching versus linear time temporal logic, Journal of the Association of Computing Machinery 33 (1) (1986) 151–178.

[30] M. Huth, M. Kwiatkowska, The semantics of fair recursion with divergence, submitted. Technical report CSR-96-7. (1996).

[31] P. Panangaden, The expressive power of indeterminate primitives in asynchronous computations, Tech. Rep. SOCS-95.8, School of Computer Science, McGill (1995).

[32] G. Plotkin, A powerdomain for countable non-determinism, in: Automata, Languages and Programming (ICALP), Ninth Colloquium, Vol. 140 of LNCS, Springer-Verlag, 1982, pp. 418–428.

[33] J. Malitz, Introduction to Mathematical Logic, Undergraduate Texts in Mathematics, Springer-Verlag, 1979.

A Grothendieck topology for a partial order

Here we give the definitions from [17] of a *Grothendieck topology* for a category P and the *sup* topology, specialised to the case where 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 38 (Grothendieck topology for a partial order) *A* Grothendieck topology for a partial order P*, is a function J which assigns to each object p of P a set $J(p)$ of sieves on p, in such a way that*

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

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 *amalgamation*, 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* amalgamation.

Definition 39 (separated presheaves and sheaves) *For a partial order* P *and a Grothendieck topology* J *on* P*, let* $\mathbf{Sp}_J(\widehat{\mathsf{P}})$ *and* $\mathbf{Sh}_J(\widehat{\mathsf{P}})$ *be the full subcategories of* $\widehat{\mathsf{P}}$ *with objects 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 α in Inf (as defined in Sec. 1), a sieve on α is simply a prefix closed set of sequences below α. We only use the *sup* topology on Inf, which to each sequence α assigns the set $\{S \mid S \text{ is a sieve on } \alpha \text{ and } \bigsqcup S = \alpha\}$, i.e. of all sieves that have α as supremum. It is easy to check that this satisfy the conditions in Def. 38, and that it works for any partial order. This topology is in fact the *canonical* topology for Inf, being the largest topology such that $\mathcal{Y}_{\mathsf{Inf}}\alpha$ is a sheaf for any α.

Definition 40 (sup topology for Inf) *For the partial order* 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 α is finite then $J(\alpha)$ contains just the maximal sieve $\alpha{\downarrow}$ on α.

B Proof of Full Abstraction

We will here give a more detailed proof outline for Prop. 36 of Sec. 6.5 as repeated below. Recall that $\mathcal{T}_g$ refer to the set of all closed terms of SCCSϵ 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 SCCSϵ with only guarded recursion.

Proposition 41 (Prop. 36 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 SCCSϵ, $FV(t)$ will denote the set of free variables in t. As in [18] [4] 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
- $gd(\mathsf{rec}\, x.t) = gd(t\lceil A) = gd(\epsilon t) = gd(t) + 1$.

This is a well defined ordinal, but not necessarily a finite number because sums can be infinite. As in [18] the following is a key property of gd for use in inductive proofs in the guard depth of terms with only guarded induction.

Lemma 42 *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 generalised transition system $T = (S, i, \rightarrow, Adm, Act)$ and $s \in S$ we define *the generalised transition system above s in T* by $T_{s\lhd} = (S_{s\lhd}, s, \rightarrow_{s\lhd}, Adm_{s\lhd}, Act)$, where

- $S_{s\lhd} = \{s' \mid s \rightarrow^* s'\}$,
- $\rightarrow_{s\lhd} = \rightarrow \cap\, (S_{s\lhd} \times Act \times S_{s\lhd})$ and
- $Adm_{s\lhd} = Adm \cap \rightarrow^\infty_{s\lhd}$.

For any term t in $\mathcal{T}_\epsilon$, let $\mathcal{D}_\epsilon(t) = (S_{d(t)}, (\bot, *), \rightarrow_t, Adm_{d(t)}, Act)$. Recall that $S_{d(t)} = \big\{(\alpha, e) \mid \alpha \in \mathsf{Inf} \text{ and } e \in [\![t]\!](\alpha)\big\}$ and $*$ is the unique element of $[\![t]\!](\bot)$. Let $\mathcal{O}_\epsilon(t) = (S_{o(t)}, t, \rightarrow, Adm_{o(t)}, Act)$. 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 $(\bot, *) \rightarrow^n s$.

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

[4] However, we use the convention from [33] that $\lambda + 1$ is the successor of λ.

Definition 43 *Let* $\mathcal{S}_T = \{(s,t) \mid s \in S_{d(t)} \text{ and } t \in \mathcal{T}_g\}$. *Define* $f\colon \mathcal{S}_T \to \mathcal{T}_g$ *by well founded recursion as follows (writing* $f_t(s)$ *for* $f(s,t)$*)*

- $f_t(\bot, *) = t$,
- $f_{a:t}(a\alpha, e) = f_t(\alpha, e)$,
- $f_{\Sigma_{i\in I} t_i}\Big(\alpha, (\mathsf{sum}\ i, s)\Big) = f_{t_i}(s)$,
- $f_{t_1 \times t_2}(\alpha, s_1 \times s_2) = f_{t_1}(s_1) \times f_{t_2}(s_2)$,
- $f_{\mathsf{rec}\, x.t}(s) = f_{t[\mathsf{rec}\, x.t/x]}\Big(\mathcal{E}l(\rho_t)s\Big) \quad$ *if* $h(s) > 0$,
- $f_{t\restriction A}(s) = f_t(s)\restriction A \quad$ *if* $h(s) > 0$,
- $f_{\epsilon_n t}\Big(1^{n'}, (\mathsf{del}\ n', (\bot, *))\Big) = \epsilon_{n+n'} t$,
- $f_{\epsilon_n t}\Big(1^{n'}\alpha, (\mathsf{del}\ n', s)\Big) = f_t(s) \quad$ *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* $\mathcal{S}_T$ *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 $f\colon \mathcal{S}_T \to \mathcal{T}_g$ 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 44 *Let* $\mathcal{F} = \{f_t\colon S_{d(t)} \to \mathcal{T}_g \mid t \in \mathcal{T}_g\}$ *be the collection of maps given above. Then there exists a collection of isomorphisms of generalised synchronisation trees* $\{\sigma_{t,s}\colon \mathcal{D}_\epsilon(t)_{s\lhd} \to \mathcal{D}_\epsilon\Big(f_t(s)\Big) \mid t \in \mathcal{T}_g \text{ and } s \in S_{d(t)}\}$ *such that if* $s \to_t^* s'$ *in* $\mathcal{D}_\epsilon(t)$ *then*

$$(f_t(s) = t') \implies f_t(s') = f_{t'}\Big(\sigma_{t,s}(s')\Big), \tag{B.1}$$

Proof (Sketch) We proceed by induction in the height of the states s. First we define $\sigma_{t,s}\colon \mathcal{D}_\epsilon(t)_{s\lhd} \to \mathcal{D}_\epsilon\Big(f_t(s)\Big)$ for $t \in \mathcal{T}_g$ and $s = (\bot, *) \in S_{d(t)}$, i.e. for all roots. Then $\mathcal{D}_\epsilon(t)_{s\lhd} = \mathcal{D}_\epsilon(t)$ and $f_t(s) = t$ so we can define $\sigma_{t,s} = 1_{\mathcal{D}_\epsilon(t)}$. We then define $\sigma_{t,s}\colon \mathcal{D}_\epsilon(t)_{s\lhd} \to \mathcal{D}_\epsilon\Big(f_t(s)\Big)$ 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 $\mathcal{D}_\epsilon(t)_{s\lhd}$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 $f_t(s') = f_{t''}\Big(\sigma_{t,s_n}(s')\Big) = f_{t'}\Big(\sigma_{t'', \sigma_{t,s_n}(s)}(\sigma_{t,s_n}(s'))\Big) = f_{t'}\Big(\sigma_{t,s}(s')\Big)$. $\square$

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

Lemma 45 *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. 43 above. If* $f_t(s_0) = t_0$ *for* $s_0 \in S_{d(t)}$ *then*

$$\left(\exists s_1 \in S_{d(t)}.s_0 \xrightarrow{a}_t s_1 \text{ and } f_t(s_1) = t_1\right) \text{ if and only if } t_0 \xrightarrow{a} t_1 \ , \qquad \text{(B.2)}$$

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

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

Then (B.2) follows for $s_0 \in S_{d(t)}$ and $f_t(s_0) = t_0$ by using (B.1) of Lem. 44. □

Corollary 46 *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* $\mathsf{Fin}_\bot$*-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 ϕ of $\mathcal{D}_\epsilon(t)$ we can always find a non-empty prefix of the image of ϕ under f_t, in which all initially waiting subagents are fulfilled.

Lemma 47 *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*

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

Proof Easy induction in $sd(t_0)$ using Lem. 44. □

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

Lemma 48 *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. 15.*

We can now conclude from Lem. 17, Cor. 46 and Lem. 48 that f_t defines a morphism of generalised transition systems.

Proposition 49 *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. We will let* $F_t\colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$ *refer to this morphism.*

To show that $F_t\colon \mathcal{D}_\epsilon(t) \to \mathcal{O}_\epsilon(t)$ is an $\mathsf{Inf}_\perp$-*open* morphism we need to check the two zig-zag conditions of Prop. 25 in Sec. 4. As already mentioned above, the first condition follows directly from Lem. 45. 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 $Adm_{f_{d(t)}} = f_{t\infty}^{-1}(Adm_{o(t)}) = \{\phi \in Comp\Big(\mathcal{D}_\epsilon(t)\Big) \mid f_{t\infty}(\phi) \in Adm_{o(t)}\}$. 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 SCCSϵ and let $\mathcal{T}_g^\top = \mathcal{T}_g \cup \{\top\}$. 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 $\sigma_{\top,s}\colon \mathcal{D}_\epsilon(\top)_{s\lhd} \to \mathcal{D}_\epsilon(f_\top(s))$ for $s \in S_{d(\top)}$ extending Def. 43 and Lem. 44 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 50 *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 if*

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

then

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

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

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

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

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

The well founded order is, as in Def. 43, given by $(s_1, t_1) < (s_2, t_2)$ if $h(s_1) < h(s_2)$ or $\Big(h(s_1) = h(s_2) \wedge gd(t_1) < gd(t_2)\Big)$. $\Box$

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

Corollary 51 *Let $t' \in \mathcal{T}_g$ and $t \in \mathcal{T}_g^o$ such that $FV(t) = \{x\}$ and let $m\colon [\![t']\!] \to [\![\top]\!]$ be the unique morphism to the terminal presheaf. Then*

$\forall \phi \in Comp(\mathcal{O}_\epsilon(t[t'/x])),$
$\quad f_{t[\top/x]_\infty}(\mathcal{E}l([\![t]\!]m)_\infty\phi)$ *is inadmissible* $\implies$ $f_{t[t'/x]_\infty}(\phi)$ *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 52 *Let t be a standard term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$ and let $\rho_t\colon [\![\mathsf{rec}\, x.t]\!] \to [\![t[\mathsf{rec}\, x.t/x]]\!]$ be the isomorphism given in Sec. 6.2. Then $\forall n \in \omega, \forall \phi \in Comp(\mathcal{O}_\epsilon(t^n[\mathsf{rec}\, x.t/x]))$,*

$f_{t^{n+1}[\mathsf{rec}\, x.t/x]_\infty}(\mathcal{E}l([\![t]\!]^n\rho_{t_1})_\infty\phi)$ *is inadmissible* $\implies$ $f_{t^n[\mathsf{rec}\, x.t/x]_\infty}(\phi)$ *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 $\forall p \in Pos, f_{t[\mathsf{rec}\, x.t/x]}(\bot, *)p = \epsilon_n u \implies n = 0$, so we get that $\forall s \in S_{d(t[\mathsf{rec}\, x.t/x])}\forall p \in Pos \forall n > 1, f_{t[\mathsf{rec}\, x.t/x]}\Big(\mathcal{E}l(\rho_t)s\Big)p = \epsilon_n u \implies f_{\mathsf{rec}\, x.t}(s)p = \epsilon_n u$ and the desired result follows from Lem. 50 and Def. 30, by noting that $t^{n+1}[\mathsf{rec}\, x.t/x] = t^n[t[\mathsf{rec}\, x.t/x]/x]$ and $[\![t]\!]^n = [\![t^n]\!]$. $\Box$

Lemma 53 *Let t be a term in $\mathcal{T}_g^o$ such that $FV(t) = \{x\}$. Then*

$\forall t' \in \mathcal{T}_g^\top, Adm_{f_{d(t')}} \subseteq Adm_{d(t')} \implies Adm_{f_{d(t[t'/x])}} \subseteq Adm_{d(t[t'/x])}$

implies

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

Proof By an easy induction in n. $\Box$

Proposition 54 *Let t be a standard term in $\mathcal{T}_g^o$ such that $FV(t) \subseteq \{x\}$. Then $\forall t' \in \mathcal{T}_g^\top, (Adm_{f_{d(t')}} \subseteq Adm_{d(t')} \implies Adm_{f_{d(t[t'/x])}} \subseteq Adm_{d(t[t'/x])})$.*

Proof (Sketch) By structural induction in t, using Lem. 52, Lem. 51 and Lem 53 above in the case for recursion. $\Box$

From the proposition above it follows that $f_{t\infty}$ reflects admissibility for any closed term t, which was what we wanted to show.

Corollary 55 *Let t be a standard term in $\mathcal{T}_g$. Then $Adm_{f_{d(t)}} \subseteq Adm_{d(t)}$.*