# Categorical Models for Fairness: Completion vs Delay

Thomas T. Hildebrandt[*]

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

## Abstract

We study two different approaches to semantics for fairness within the categorical framework of presheaf models for concurrency $[1, 2]$. The first approach, used in e.g. dataflow models, is based on representing finite or infinite *completed* observations. A completed observation is the result of an *infinite* computation. A *finite* completed observation is then the result of an infinite computation that only produces finitely many actions, i.e. at some point it stops producing output but stays active (as opposed to being deadlocked). The second approach is that of e.g. Milner's SCCS with finite delay [6], having an observable *delay* action and representing also finite incomplete observations, but only infinite completed observations.

More precisely, we consider (separated) presheaves over respectively the category

$$\mathsf{Comp} = \big(pf(Act^{\omega} \cup Act^{*}\circlearrowright), \leq_{pf}\big),$$

of *incomplete finite, and completed infinite and finite observations,* and the category

$$\mathsf{Inf} = \big(pf((Act \cup \{1\})^{\omega}), \leq_{pf}\big),$$

of *incompleted finite and completed infinite observations, possibly containing delays*, where $Act$ is some set of actions, such that $1, \circlearrowright \notin Act$. We refer to $1$ as the *delay* action and $\circlearrowright$ as the *idle completion*. For $S$ a set of sequences, $pf(S)$ is the set of (finite and infinite) prefixes of sequences in $S$ and $\leq_{pf}$ is the usual prefix order. The latter model was applied previously by the author $[5, 4]$ to give a denotational semantics of SCCS with finite delay shown to be fully abstract with respect to Hennessy and Stirlings extended bisimulation [3]. One of the motivations for studying finite delay is the ability to encode a fair asynchronous parallel operator. However, the model contains a lot more processes than just those obtained from the encoding. Moreover, the introduction of delay actions makes the modelling of synchronous communication less obvious. This raises the questions of identifying the behaviours of fair processes and giving a "delay free" representation of the processes, both answered by the present paper. We apply the first model to give a denotational semantics for a CSP-like process language, FCSP, with a fair asynchronous parallel operator with synchronous communication, an idle process (the identity for the fair parallel operator) and a deadlocked process (represented by the incomplete observation with no actions). The canonical notion of bisimulation obtained from span of open maps is sensitive to both fairness and deadlock.

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

We give a syntactic encoding of FCSP into SCCS with finite delay. Benefiting from the categorical presentation we show that, semantically, the encoding corresponds to the inverse image of the geometric morphism induced by the functor $\gamma\colon \mathsf{Inf} \to \mathsf{Comp}$ which removes delay actions and adds the idle completion, if the sequence contained infinitely many delay actions but only finitely many non-delay actions.

## References

1. G. L. Cattani. *Presheaf models for concurrency*. PhD thesis, Aarhus University, 1999.
2. G. L. Cattani and G. Winskel. Presheaf models for ccs-like languages. *Submitted*, 1999.
3. M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, pages 23–52, 1985.
4. T. T. Hildebrandt. *Categorical Models for Concurrency: Independence, Fairness and Dataflow*. PhD thesis, Department of Computer Scence, University of Aarhus, Denmark, October 1999.
5. T. T. Hildebrandt. A fully abstract presheaf semantics of SCCS with finite delay. Research Series RS-99-28, BRICS, Department of Computer Science, University of Aarhus, Sept. 1999. Appears in Proceedings of CTCS'99, ENTCS, Vol. 29 and in a revised version submitted to TCS.
6. R. Milner. A finite delay operator in synchronous ccs. Technical Report 116-82, University of Edinburgh, Dept. of Computer Science, Kings Buildings, 1982.