Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 325 (2016) 237–252

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Reducing Complex CSP Models to Traces via Priority

David Mestel[1] A.W. Roscoe[2]

Department of Computer Science, University of Oxford

Abstract

Hoare's Communicating Sequential Processes (CSP) [6] admits a rich universe of semantic models. In this paper we study finite observational models, of which at least six have been identified for CSP, namely traces, failures, revivals, acceptances, refusal testing and finite linear observations [11]. We show how to use the recently-introduced *priority* operator ([12], ch.20) to transform refinement questions in these models into trace refinement (language inclusion) tests. Furthermore, we are able to generalise this to any (rational) finite observational model. As well as being of theoretical interest, this is of practical significance since the state-of-the-art refinement checking tool FDR3 [4] currently only supports two such models.

Keywords: CSP, denotational semantics, priority

1 Introduction

A number of different forms of process calculus have been developed for the modeling of concurrent programs, including Hoare's Communicating Sequential Processes (CSP) [6], Milner's Calculus of Communicating Systems (CCS) [7], and the π-calculus [8]. Unlike the latter two, CSP's semantics are traditionally given in behavioural semantic models coarser than bisimulation.

In this paper, we study finite linear-time observational models for CSP; that is, models where all observations considered can be determined in a finite time by an experimenter who can see the visible events a process communicates and the sets of events it can offer in any stable state. While the experimenter can run the process arbitrarily often, he or she can only record the results of individual finite executions. Thus each behaviour recorded can be deduced from a single finite sequence of events together with the sets of events accepted in stable states during and immediately after this *trace*.

[1] Email: david.mestel@cs.ox.ac.uk

[2] Email: bill.roscoe@cs.ox.ac.uk

http://dx.doi.org/10.1016/j.entcs.2016.09.041

At least six such models have been considered for CSP, but the state-of-the art refinement checking tool, FDR3 [4], currently only supports two, namely *traces* and *failures* (it also supports the failures-divergences model, which is not finite observational).

We present a construction which produces a context $\mathcal{C}$ such that refinement questions in the failures model correspond to trace refinement questions under the application of $\mathcal{C}$. We are able to generalise this to show (Theorem 5.4) that a similar construction is possible not only for the six models which have been studied, but also for any sensible finite observational model (where 'sensible' means that the model can be recognised by a finite-memory computer, in a sense which we shall make precise).

We first briefly describe the language of CSP. We next give an informal description of our construction for the failures model. To prove the result in full generality, we first give a formal definition of a finite observational model, and of the notion of rationality. We then describe our general construction. Finally we discuss performance and optimisation issues.

2 The CSP language

We provide a brief outline of the language, largely taken from [11]; the reader is encouraged to consult [12] for a more comprehensive treatment.

Throughout, Σ is taken to be a finite nonempty set of communications that are visible and can only happen when the observing environment permits via handshaken communication. The actions of every process are taken from $\Sigma \cup \{\tau\}$, where τ is the invisible internal action that cannot be prevented by the environment. Note that the usual treatment of CSP permits sequential composition by including another un-preventable event $\checkmark$ to represent termination; this adds slight complications to each model and we omit it for simplicity. It could be added back without any significant alteration to the results of this paper.

The constant processes of CSP are

- *STOP* which does nothing—a representation of deadlock.
- **div** which performs (only) an infinite sequence of internal τ actions—a representation of divergence or livelock.
- *CHAOS* which can do anything except diverge.

The prefixing operator introduces communication:

- $a \to P$ communicates the event a before behaving like P.

There are two forms of binary choice between a pair of processes:

- $P \sqcap Q$ lets the process decide to behave like P or like Q: this is *nondeterministic* or *internal* choice.
- $P \,\Box\, Q$ offers the environment the choice between the initial Σ-events of P and Q. If the one selected is unambiguous then it continues to behave like the one chosen; if it is an initial event of both then the subsequent behaviour is nondeterministic.