A Corrected Failure-Divergence Model for CSP in Isabelle/HOL

H. Tej, B. Wolff

Universität Bremen, FB3
Postfach 330440
D-28334 Bremen
{bu,ht}@informatik.uni-bremen.de

Abstract. We present a failure-divergence model for CSP following the concepts of [BR 85]. Its formal representation within higher order logic in the theorem prover Isabelle/HOL [Pan 94] revealed an error in the basic definition of CSP concerning the treatment of the termination symbol tick.

A corrected model has been formally proven consistent with Isabelle/HOL. Moreover, the changed version maintains the essential algebraic properties of CSP. As a result, there is a proven correct implementation of a "CSP workbench" within Isabelle.

1 Introduction

In his invited lecture at FME'96, C.A.R. Hoare presented his view on the status quo of formal methods in industry. With respect to formal proof methods, he ruled that they "are now sufficiently advanced that a [...] formal methodologist could occasionally detect [...] obscure latent errors before they occur in practice" and asked for their publication as a possible "milestone in the acceptance of formal methods" in industry.

In this paper, we report of a larger verification effort as part of the UniForM project [Kri+95]. It revealed an obscure latent error that was not detected within a decade. It can not be said that the object of interest is a "large software system" whose failure may "cost millions", but it is a well-known subject in the center of academic interest considered foundational for several formal methods tools: the theory of the failure-divergence model of CSP ([Hoa 85], [BR 85]). And indeed we hope that this work may further encourage the use of formal proof methods at least in the academic community working on formal methods.

Implementations of proof support for a formal method can roughly be divided into two categories. In direct tools like FDR [For 95], the logical rules of a method (possibly integrated into complex proof techniques) are hard-wired into the code of their implementation. Such tools tend to be difficult to modify and to formally reason about, but can possess enviable automatic proof power in specific problem domains and comfortable user interfaces.

1 This work has been supported by the German Ministry for Education and Research (BMBF) as part of the project UniForM under grant No. FKZ 01 IS 521 B2.
The other category can be labelled as logical embeddings. Formal methods such as CSP or Z can be logically embedded into an LCF-style tactical theorem prover such as HOL [GM 93] or Isabelle [Pau94]. Coming with an open system design going back to Milner, these provers allow for user-programmed extensions in a logically sound way. Their strength is flexibility, generality and expressiveness that makes them to symbolic programming environments.

In this paper we present a tool of the latter category (as a step towards a future combination with the former). After a brief introduction into the failure divergence semantics in the traditional CSP-literature, we will discuss the revealed problems and present a correction. Although the error is not "mathematically deep", it stings since its correction affects many definitions. It is shown that the corrected CSP still fulfils the desired algebraic laws. The addition of fixpoint-theory and specialised tactics extends the embedding in Isabelle/HOL to a formally proven consistent proof environment for CSP. Its use is demonstrated in a final example.

2 The Failure Divergence Semantics

In this section, we follow closely the presentation of [Cam 91], whose contribution is a formal, machine-assisted version of a subset of CSP based on [BR 85] and [Ros 88] without the sequential operator, the parallel interleave operator and a proof-theory based on fixpoint induction. With [Cam 91], we share some major design decisions, in particular the choice of the alternative process ordering in [Ros 88] (see below).

In its trace semantics model it is not possible to describe certain concepts that commonly arise when reasoning about concurrent programs. In particular, it is not possible to express non-determinism, or to distinguish deadlock from infinite internal activity. The failure-divergence model incorporates the information available in the trace-semantics, and in addition introduces the notions of refusal and divergence to model such concepts.

Example 2.1: Non-Determinism

Let $a$ and $b$ be any two events in some set of events $\Sigma$. The two processes

$$(a \rightarrow \text{Stop}) \boxempty (b \rightarrow \text{Stop}) \quad (1)$$

and

$$(a \rightarrow \text{Stop}) \sqcap (b \rightarrow \text{Stop}) \quad (2)$$

cannot be distinguished under the trace semantics, in which both processes are capable of performing the same sequences of events, i.e. both have the same set of traces \{(),(a),(b)\}. This is because both processes can either engage in $a$ and then $\text{Stop}$, or engage in $b$ and then $\text{Stop}$. We would, however, like to distinguish between a deterministic choice of $a$ or $b$ (1) and a non-deterministic choice of $a$ or $b$ (2).

This can be done by considering the events that a process can refuse to engage in when these events are offered by the environment; it cannot refuse either, so we say its maximal refusal set is the set containing all elements of $\Sigma$ other than $a$ and $b$, written $\Sigma\\{a,b\}$, i.e. it can refuse all elements in $\Sigma$ other than $a$ or $b$. In the case of the non-deterministic process (2), however, we wish to express that if the environment offers the event $a$ say, the process non-deterministically chooses either to engage in $a$,