Of wlp and CSP

Steve Dunne[1]

School of Computing
University of Teesside, Middlesbrough, UK

Abstract

We extend Morgan's well-known derivation of the Failures-Divergences semantics of an action system endowed with a wp sequential semantics, by showing how various other CSP semantics can be extracted from an action system endowed with an appropriate sequential semantics. In doing so we expose the close but hitherto largely overlooked correspondence between the various CSP semantic models and their sequential correctness counterparts.

Keywords: CSP, wlp, intermediate correctness, stable failures, CFFD

1 Communicating Sequential Processes

CSP or *Communicating Sequential Processes* [6,15] is a well-known process algebra typifying the *event-based* approach to concurrency, in which a process is characterised entirely by its externally observable possible patterns of interaction with its environment via shared primitive events drawn from a specified alphabet of possible such events. Such a purely behavioural characterisation abstracts away from any operational notion of a process as an evolving entity with an evolving internal state which at any given point determines how the process will react to the next stimulus from its environment; rather, it provides a "God's eye view" of the process whereby all its potential patterns of behaviour are simultaneously exhibited in their entirety. What particular features of this behaviour are actually observed depends on the purposes of the observer. In CSP the observer is deemed to be concerned by safety and liveness properties of his system, so his observations are therefore confined to any or all of the following:

- *traces*, a trace being any particular sequence of events in which the process is observed successively to engage;
- *failures*, a failure being a trace leading to a *refusal set*, *i.e.* a given set of events simultaneously proposed by the environment in each of which the process refuses

[1] Email: s.e.dunne@tees.ac.uk

to engage;

- *divergences*, a divergence being a trace after which the process becomes unstable by being as it were *livelocked* in an endless succession of hidden internal events so that no further meaningful observation of trace or refusal behaviour is possible.

If Σ is the event alphabet of the process concerned then Σ^* denotes the set of all finite traces on Σ, *i.e.* the set of all finite sequences of events in Σ including the empty trace $\langle\rangle$. Each failure of a process is characterised by an ordered pair (tr, X) where tr is a finite trace and X is a refusal, *i.e.* a set of events simultaneously refused by the process. Hence the failures of process collectively comprise a relation of type $\Sigma^* \leftrightarrow \mathbb{P}\,\Sigma$, known as the *failures relation* of that process. In this paper we restrict our attention to (at most) finitely nondeterministic processes, so we don't need to consider infinite traces.

1.1 CSP semantic models

There are several recognised semantic models in the CSP literature, each of which induces its own particular congruence over process terms of the CSP language, a congruence being an equivalence relation which is compositional with respect to the language operators. Each of these congruences is fully abstract with respect to some characteristic simple but significant operational test which usefully distinguishes processes in some way, see [15][Thm 9.3.1]. For describing these models it is useful to introduce at this point the two primitive CSP processes Stop, which deadlocks immediately without diverging, and Div which immediately diverges.

- **Traces** (TR): this is the simplest of our recognised models. In it each process P is denoted by its set of traces $traces(P)$. This model completely ignores divergent behaviour, even to the extent that it equates Div with Stop. It is sufficient for reasoning about safety, *i.e.* whether any given event *can* occur, but not about liveness, *i.e.* in what circumstances any given event *must* occur.
- **Failures-Divergences** (FD): this is the *de facto* standard semantic model for finitely nondeterministic CSP. In contrast to TR it takes a drastic view of divergence which regards a process as utterly unpredictable, and therefore capable of any behaviour, once divergence has occurred. Thus FD interprets Div as the least reliable of all processes, with the consequence that whenever divergence is even a possibility any other specific behaviour associated with the process at that point is completely occluded by all the possible behaviours associated with divergence. In this model each finitely nondeterministic process P is characterised by its divergence-augmented failures relation $failures_\perp(P)$, and its extension-closed set of divergences $divergences_\perp(P)$. The domain of P's divergence-augmented failures relation $failures_\perp(P)$ is denoted by $traces_\perp(P)$. This comprises the proper traces $traces(P)$ of P together with P's extension-closed divergences $divergences_\perp(P)$.
- **Stable Failures** (SF): although less prominent in the CSP literature than FD, this model is theoretically and practically significant as the weakest congruence which respects deadlock [18]. Although not quite as indifferent to divergence as TR, to quote from [16] SF certainly "turns something of blind eye" to divergence, and so, in contrast to FD, permits other specific behaviour the process may

possess alongside the divergence to be discerned. Although it deems Stop and Div to have the same traces, in fact just the empty trace $\langle\rangle$, it distinguishes between them on the basis of their refusals, Stop initially refusing everything, but Div having no refusals since it never even achieves a stable initial state from which to refuse anything. The properties of SF are analysed in detail by Roscoe in [15], although Valmari [20] attributes its origins back as far as [2]. In the SF model each process P is characterised by its traces $traces(P)$ and its stable failures, *i.e.* its ordinary non-divergence-augmented failures $failures(P)$.

- **Chaos-free Failures Divergences** (CFFD): this model was originally introduced by Valmari [20] for finite-state labelled transition systems, but then later extended by him to encompass infinite-state labelled transition systems too [19]. As Valmari explains in [19], his CFFD equivalence is "the weakest congruence that preserves deadlock and formulae written in a popular linear-time temporal logic". In the original finite-state version of the model each process P is characterised by its stable failures $failures(P)$, its minimal (*i.e.* non-extension-closed) divergences $divergences(P)$, and its initial stable states $stable(P)$. However, if only CSP operators are used then the initial stable states $stable(P)$ can be discarded. Since in this paper we consider only finitely nondeterministic CSP processes, each process P is therefore characterised here by its minimal, *i.e.* non-extension-closed, divergences $divergences(P)$ and its stable failures $failures(P)$.

1.2 Refinement relationships between the models

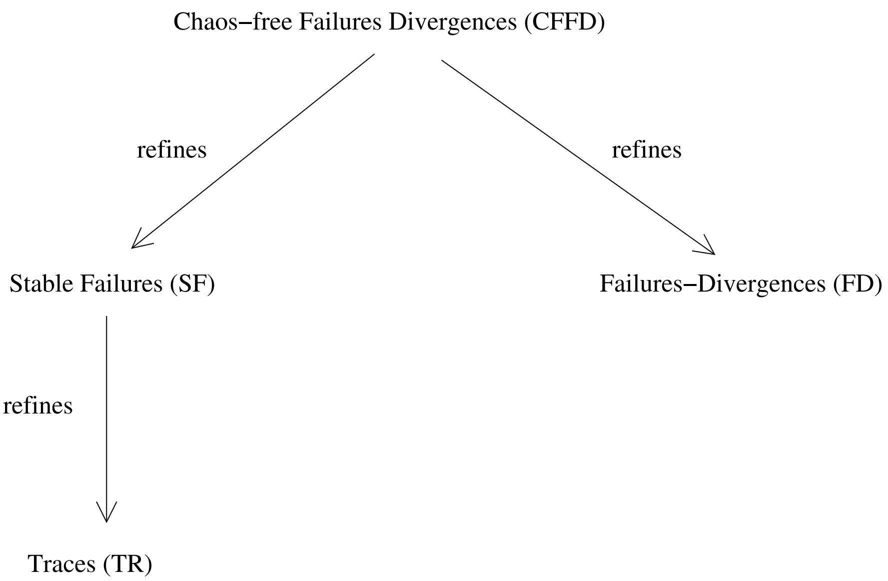

Fig. 1. Refinement Graph of the CSP semantic models

Figure 1 depicts the refinement relationships between the four CSP semantic models above. Generally speaking the more refined the model the more distinctions it makes between processes. Any of these semantic models is said to *refine* another if it makes a least as many distinctions between processes as the latter. Thus any pair of processes which are distinguished by the coarser model being refined must also be distinguished by the refining model, while conversely any pair of processes equated by the refining model must also be equated by the coarser model being refined.

Not all pairs of models are refinement-comparable. For example, SF and FD are incomparable, since each makes distinctions between some processes which the other equates. Neither does FD refine TR: although it is tempting to think that in FD the traces of a process can be extracted as the domain of its failures relation, in fact this is not so because the traces so obtained, unlike those in TR, will include all finite extensions of minimal divergences.

As can be seen in Figure 1, the most refined of our models is CFFD since this can distinguish between any processes which any of the other models can. For example CFFD distinguishes between all three of the process Stop, Div and $\mathsf{Stop} \sqcap \mathsf{Div}$, whereas SF equates $\mathsf{Stop} \sqcap \mathsf{Div}$ with Stop while in contrast FD equates $\mathsf{Stop} \sqcap \mathsf{Div}$ with Div. The actual distinctions are shown in Table 1. [2]

P	Stop	Div	$\mathsf{Stop} \sqcap \mathsf{Div}$
$traces(P)$	$\{\langle\rangle\}$	$\{\langle\rangle\}$	$\{\langle\rangle\}$
$failures(P)$	$\{(\langle\rangle, X) \mid X \subseteq \Sigma\}$	$\{\}$	$\{(\langle\rangle, X) \mid X \subseteq \Sigma\}$
$divergences(P)$	$\{\}$	$\{\langle\rangle\}$	$\{\langle\rangle\}$
$divergences_\perp(P)$	$\{\}$	Σ^*	Σ^*
$failures_\perp(P)$	$\{(\langle\rangle, X) \mid X \subseteq \Sigma\}$	$\Sigma^* \times \mathbb{P}\,\Sigma$	$\Sigma^* \times \mathbb{P}\,\Sigma$

Table 1

2 State-based Representations of Processes

A duality has long been recognised between the purely behavioural description of a concurrent process discussed in Section 1 and its corresponding more operational state-based description, for example as a labelled transition system (LTS) [21], action system [1] or abstract data type (ADT) [3,17]. The first explicit state-based characterisations of a behavioural process to attract widespread attention were that of Josephs [11] describing how a non-divergent CSP process could be represented as an LTS, and that of He [8] who developed a similar state-based relational process model which did accommodate divergence. The motivation in both cases was to provide a state-based process model in respect of which it would be possible to adapt He *et al*'s ADT refinement proof methods of forward and backward simulation [10] to the failures-divergences refinement of processes.

Soon afterwards Morgan [12] showed how to interpret an action system as a CSP process by describing precisely how to derive its failures and divergences from its wp semantics. What was particularly striking about Morgan's approach is the way the phenomenon of divergence is handled so uniformly by it; the wp semantics gives us divergence at no extra cost, so to speak. Before we explain Morgan's method in

[2] Since we are not primarily concerned here with the termination of CSP processes, we have simplified things slightly by ignoring the special event ✓ which signals termination of a CSP process.

detail it is convenient to define several subsidiary concepts and constructs derived from the wp predicate transformer in the next subsection.

2.1 Some subsidiary concepts and constructs derived from wp

We start with an action system A endowed with a total-correctness semantics, more specifically a weakest precondition (wp) predicate-transfomer semantics. A therefore possesses a state S together with both an initialisation *init* and a repertoire of actions a_i $(i \in I)$ on S which can *fire* individually when *enabled* and modify S. The initialisation *init* and each of the actions a_i is characterised as a wp predicate transformer. We will describe a predicate transformer as *drastic* if it maps all predicates to either of the extreme predicates true or false. The initialisation *init* of an action system is always characterised as a drastic wp predicate transformer, and it must be fully *feasible*, *i.e.* strict with respect to false, so that it is always enabled.

The sequential composition $a_1 \,;\, a_2$ of two actions is defined by

$$\mathrm{wp}(a_1 \,;\, a_2 \;,\, q) \quad =_{df} \quad \mathrm{wp}(a_1 \;,\, \mathrm{wp}(a_2 \;,\, q)) \qquad \text{seq comp}$$

For convenience we introduce the notion of a conjugate wp predicate transformer, denoted cwp, where given an action a and a postcondition q we have

$$\mathrm{cwp}(a \;,\, q) \quad =_{df} \quad \neg\, \mathrm{wp}(a \;,\, \neg\, q) \qquad \text{conjugate wp}$$

We note that whereas wp is positively conjunctive for demonically nondeterministic programs since it distributes through all non-empty conjunctions of postconditions, cwp is positively *disjunctive* since it distributes through all non-empty disjunctions of postconditions.

We also define the following characteristic predicates of an action a

$$\begin{array}{llll} \mathrm{trm}(a) & =_{df} & \mathrm{wp}(a \;,\, \mathrm{true}) & \text{termination} \\ \mathrm{fis}(a) & =_{df} & \mathrm{cwp}(a \;,\, \mathrm{true}) & \text{feasibility guard} \end{array}$$

We extend the domain of fis in an obvious way to sets of actions:

$$\mathrm{fis}(X) \quad =_{df} \quad \bigvee_{a \in X} \mathrm{fis}(a) \qquad \text{overall feasibility}$$

In particular, since the empty disjunction is trivially false, we have $\mathrm{fis}(\{\}) = \mathrm{false}$.

2.2 Morgan's behavioural interpretation of an action system

We now describe how Morgan [12] extracts the FD semantics of the action system A endowed with a wp semantics.

- A sequence $\langle a_1, a_2 \ldots a_n \rangle$ of actions is a **trace** of action system A precisely if $\mathrm{fis}(\mathit{init} \,;\, a_1 \,;\, a_2 \,;\, \ldots \,;\, a_n)$ holds.
- A sequence $\langle a_1, a_2 \ldots a_n \rangle$ of actions is a **divergence** of action system A precisely if $\neg\, \mathrm{trm}(\mathit{init} \,;\, a_1 \,;\, a_2 \,;\, \ldots \,;\, a_n)$ holds.
- A pair $(\langle a_1, a_2, \ldots, a_n \rangle \;, X)$, where $\langle a_1, a_2, \ldots, a_n \rangle$ is a sequence of actions and X is a set of actions, is a **failure** of action system A precisely if $\mathrm{cwp}(\mathit{init} \,;\, a_1 \,;\, a_2 \,;\, \ldots \,;\, a_n \;,\, \neg\, \mathrm{fis}(X))$ holds.

Note that the drastic nature of A's initialisation *init* ensures that all three conditions above are absolute rather than contingent. That is, each of them is either wholly true or wholly false.

2.3 Limitation of total correctness

Given the striking mathematical elegance and conciseness of Morgan's method of extracting the FD semantics of an action system described in the previous subsection, the question naturally arises of whether a semantics corresponding to any of the other CSP semantic models we described in Section 1 can be similarly extracted from the same action system. In fact the answer is no, as long as the wp semantics with which our action system in Section 2.2 is endowed is a *total-correctness* semantics. Such a semantics doesn't provide a rich enough representation of the actual operational behaviour of the action system to enable us to extract its denotations in any of the other semantic models TR, SF or CFFD.

As we will see, to extract the TR semantics of an action system we require at least a partial-correctness specification of all its actions, while to extract its CFFD semantics we need a full general-correctness specification of all its actions. Even more interestingly, to extract the SF semantics of an action system it turns out we require a specification of all its actions in a form of correctness strictly intermediate between partial and general correctness, for which as far as we are aware no-one has yet coined a name. For want of a better name, therefore, we call this simply *intermediate correctness*. We will describe all these various concepts of correctness for sequential computations in detail in the next section.

3 Sequential Correctness Concepts

When Dijkstra [4] invented his weakest-precondition (wp) predicate-transformer semantics he also introduced (but seems at first to have made little use of) a second predicate transformer which he called a *weakest-liberal-precondition* (wlp) predicate transformer; while wp provides a *total-correctness* semantics, wlp in contrast provides a *partial-correctness* semantics. Unlike wp, wlp is strict with respect to true, *i.e.* for any program a we have $\mathrm{wlp}(a\ ,\mathrm{true}) = \mathrm{true}$. So whereas wp is positively conjunctive for demonically nondeterministic programs, wlp is universally conjunctive for demonically nondeterministic programs since it also distributes through the vacuously true empty conjunction. In combination wp and wlp provide a *general-correctness* semantics [7,14,5]. The two are linked by the following rule

$$\mathrm{wp}(a\ ,q) \quad = \quad \mathrm{wp}(a\ ,\mathrm{true}) \wedge \mathrm{wlp}(a\ ,q) \qquad \text{wp-wlp linkage}$$

Since $\mathrm{wp}(a\ ,\mathrm{true})$ is also denoted by $\mathrm{trm}(a)$ we can exploit this linkage rule by choosing to deem trm and wlp as the fundamental components of our general-correctness semantics while relegating wp to the status of a derived component:

$$\mathrm{wp}(a\ ,q) \quad =_{df} \quad \mathrm{trm}(a) \wedge \mathrm{wlp}(a\ ,q) \qquad \text{wp definition}$$

3.1 Some subsidiary concepts and constructs derived from trm and wlp

For convenience we also introduce the notion of conjugate wlp predicate transformer, denoted by cwlp, where given an action a and a postcondition q we have

$$\mathrm{cwlp}(a\ ,q) \quad =_{df} \quad \neg\ \mathrm{wlp}(a\ ,\neg\ q) \qquad \text{conjugate wlp}$$

We note that cwlp is universally disjunctive for demonically nondeterministic programs.

We also define the following further characteristic predicates of an action a in the context of general correctness:

$$\mathrm{fec}(a) \quad =_{df} \quad \mathrm{cwlp}(a\ ,\mathrm{true}) \qquad \text{fecundity guard}$$

$$\mathrm{fit}(a) \quad =_{df} \quad \mathrm{trm}(a) \wedge \mathrm{fec}(a) \qquad \text{fitness guard}$$

We extend the domain of fec in the obvious way to sets of actions:

$$\mathrm{fec}(X) \quad =_{df} \quad \bigvee_{a \in X} \mathrm{fec}(a) \qquad \text{overall fecundity}$$

Again, since the empty disjunction is trivially false, we have in particular that $\mathrm{fec}(\{\}) = \mathrm{false}$.

3.2 Varieties of sequential correctness

We are now in a position to characterise the various sequential correctness concepts we will subsequently need:

wlp	partial correctness
wp	total correctness
wlp , trm	general correctness
wlp , fit	intermediate correctness

The relationship between these correctness concepts is depicted in the graph shown in Figure 2. The astute reader will notice the close resemblance in shape between this graph and the one in Figure 1. This resemblance is no accident.

In Figure 2's refinement graph intermediate correctness sits strictly between general correctness and partial correctness because the fitness guard $\mathrm{fit}(a)$ of an action a is derivable from its termination condition $\mathrm{trm}(a)$ and its wlp semantics, whereas the reverse is not the case: that is to say, the termination condition $\mathrm{trm}(a)$ of an action a is not derivable from its fitness guard $\mathrm{fit}(a)$ and its wlp semantics.

Similarly, total correctness sits on a different branch of same the refinement graph altogether from intermediate correctness and partial correctness because the wlp semantics of an action a cannot be derived from its wp semantics, and nor can the wp semantics of an action a be derived from its wlp semantics [13].Indeed, even if we supplement the wlp semantics of an action a with its fitness guard $\mathrm{fit}(a)$, we still cannot derive its wp semantics.

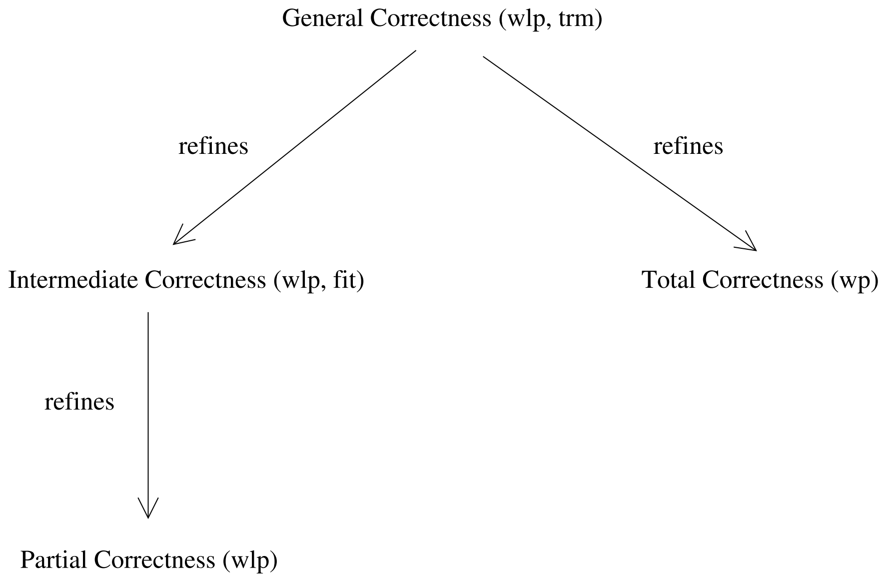

Fig. 2. Relationship between sequential correctness concepts

4 Other CSP semantics of Action Systems

In ths section we show how its TR, SF and CFFD semantics can respectively be extracted from an action system endowed with an appropriate sequential-correctness semantics. But first we introduce a further extension of our notation. If tr is a sequence of actions $\langle a_1, a_2, \ldots, a_n \rangle$ we will write $\mathrm{cwlp}(init\,;\; tr\;, q)$ to denote $\mathrm{cwlp}(init\,;\, a_1\,;\, a_2\,;\, \ldots\,;\, a_n\;, q)$. In the special case where tr is the empty sequence $\mathrm{cwlp}(init\,;\; tr\;, q)$ means $\mathrm{cwlp}(init\;, q)$. In the following subsections we can now describe in detail these extractions.

4.1 TR semantics

We start with an action system A endowed with a partial-correctness semantics, specifically a wlp semantics, and whose initialisation $init$ is a drastic wlp predicate transformer such that $\mathrm{fec}(init)$ holds. A sequence tr of actions is a **trace** of A precisely if $\mathrm{fec}(init\,; tr)$ holds.

4.2 SF semantics

We start with an action system A endowed with an intermediate-correctness semantics, specifically a (wlp, fit) semantics, and whose initialisation $init$ is a drastic wlp predicate transformer such that $\mathrm{fec}(init)$ holds. The traces and stable failures of A can be extracted as follows, noting that we separate the cases of a stable failure associated with an empty trace $\langle\rangle$, and of a non-empty trace $tr ^\frown \langle a \rangle$ for some trace tr and some action a:

- A sequence tr of actions is a **trace** of A precisely if $\mathrm{fec}(init\,; tr)$ holds.
- An ordered pair $(\langle\rangle, X)$ is a **stable failure** of A if $\mathrm{fit}(init)$ and $\mathrm{cwlp}(init\;, \neg\,\mathrm{fec}(X))$ both hold.

- An ordered pair $(tr \frown \langle a \rangle\ , X)$ is a **stable failure** of A if $\mathrm{cwlp}(init\, ;\, tr\ , (\mathrm{fit}(a) \wedge \mathrm{cwlp}(a\ , \neg\, \mathrm{fec}(X))))$ holds.

Note how all the derivations above employ only those semantic characteristics of our action system A such as the fitness guard fit which belong directly to A's intermediate-correctness semantics, or else ones such as fec and cwlp which are directly derived from wlp, itself part of A's intermediate-correctness semantics.

4.3 CFFD semantics

We start with an action system A endowed with a general-correctness semantics, specifically a (wlp, trm) semantics, and whose initialisation *init* is a drastic wlp predicate transformer such that fec(*init*) holds. The stable failures of A can be extracted in a similar way to that for the SF semantics above, although here for the sake of clarity we employ the termination condition trm rather than the fitness guard fit of our initialisation *init* and actions a. Again we separate the cases of a stable failure associated with an empty trace $\langle\rangle$ and with a non-empty trace $tr \frown \langle a \rangle$ for some trace tr and some action a. Similarly, in defining how the divergences of A are extracted we separate the cases of an empty-trace divergence $\langle\rangle$ and of a non-empty-trace divergence $tr \frown \langle a \rangle$ for some trace tr and some action a:

- An ordered pair $(\langle\rangle, X)$ is a **stable failure** of A if $\mathrm{trm}(init)$ and $\mathrm{cwlp}(init\ , \neg\, \mathrm{fec}(X))$ both hold.
- An ordered pair $(tr \frown \langle a \rangle\ , X)$ is a **stable failure** of A if $\mathrm{cwlp}(init\, ;\, tr\ , (\mathrm{trm}(a) \wedge \mathrm{cwlp}(a\ , \neg\, \mathrm{fec}(X))))$ holds.
- $\langle\rangle$ is a **divergence** of A if $\neg\, \mathrm{trm}(init)$ holds;
- $tr \frown \langle a \rangle$ is a **divergence** of A if $\mathrm{cwlp}(init\, ;\, tr\ , \neg\, trm(a))$ holds.

5 Conclusion

We have extended Morgan's derivation of the Failures-Divergences semantics of an action system endowed with a wp sequential semantics, by showing how various other CSP semantics can be extracted from an action system which is endowed in each case with an appropriate corresponding sequential semantics other than wp. In doing so we have exposed the close but perhaps hitherto largely overlooked correspondence which exists between the various CSP semantic models and their sequential-correctness counterparts.

Woodcock and Morgan [22] exploited the action-system representation of processes established in [12] to formulate wp-based sound and jointly complete proof obligations for the failures-divergences refinement of an abstract data type employing the same proof methods from [10] as Josephs [11] and He [8,9] each had done. It remains for analagous wlp-based proof obligations to be formulated fot the stable-failures refinement of an abstract data type, and indeed for its chaos-free-failures-divergences refinement too.

References

[1] R.J.R. Back and R. Kurki-Suonio. Decentralisation of process nets with centralised control. In *2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 131–142, 1983.

[2] J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Failures without chaos: A new process semantics for fair abstraction. In *Formal Description of Programming Concepts 3*, pages 77–103. North-Holland, 1987.

[3] J. Derrick and E. Boiten. *Refinement in Z and Object-Z*. Springer, 2001.

[4] E.W. Dijkstra. *A Discipline of Programming*. Prentice-Hall International, 1976.

[5] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics*. Springer Berlin, 1990.

[6] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[7] D. Jacobs and D. Gries. General correctness: a unification of partial and total correctness. *Acta Informatica*, 22:67–83, 1985.

[8] He Jifeng. Process refinement. In J. McDermid, editor, *The Theory and Practice of Refinement*. Butterworths, 1989.

[9] He Jifeng. Process simulation and refinement. *Formal Aspects of Computing*, 1(3):229–241, 1989.

[10] He Jifeng, C.A.R. Hoare, and J.W. Sanders. Data refinement refined. In B. Robinet and R.Wilhelm, editors, *Proceedings ESOP'86*, number 213 in Lecture Notes in Computer Science, pages 187–196. Springer-Verlag, 1986.

[11] M. B. Josephs. A state-based approach to communicating processes. *Distributed Computing*, 3:9–18, 1988.

[12] C.C. Morgan. Of wp and CSP. In W.H.J. Feijen, A.J.M. van Gasteren, D. Gries, and J. Misra, editors, *Beauty is our business: a birthday salute to Edsger W. Dijkstra*, pages 319–326. Springer, 1990.

[13] C.C. Morgan and A. McIver. Unifying wp and wlp. *Information Processing Letters*, 59:159–163, 1996.

[14] G. Nelson. A generalisation of Dijkstra's calculus. *ACM Transactions on Programmg Languages and Systems*, 11(4), 1989.

[15] A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1998.

[16] S. Schneider. *Concurrent and Real-time Systems: The CSP Approach*. Wiley, 2000.

[17] S. Schneider. Non-blocking data refinement and traces-divergences semantics. Technical report CS-04-09, Department of Computing. University of Surrey, 2004.

[18] A. Valmari. The weakest deadlock-preserving congruence. *Information Processing Letters*, 53:341–346, 1995.

[19] A. Valmari. A chaos-free failures divergences semantics with applications to verification. Cornerstones in Computing, pages 365–382. Palgrove, 2000.

[20] A. Valmari and M. Tienari. An improved failures equivalence for finite-state systems with a reduction algorithm. pages 3–18. North-Holland, 1991.

[21] R.J. van Glabbeek. The linear time - branching time spectrum I: the semantics of concrete sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2001.

[22] J.C.P. Woodcock and C.C. Morgan. Refinement of state-based concurrent systems. In Dines Bjørner, C. A. R. Hoare, and Hans Langmaack, editors, *VDM '90, VDM and Z - Formal Methods in Software Development, Third International Symposium of VDM Europe*, number 428 in Lecture Notes in Computer Science, pages 340–351. Springer, 1990.