

Reasoning on Responsiveness – Extending CSP-Prover by the model \mathcal{R}

D. Gift Samuel¹, Yoshinao Isobe², Markus Roggenbach^{1*}

¹ University Swansea, United Kingdom

² AIST, Tsukuba, Japan

The process algebra CSP [4, 11, 15, 1] provides a well-established, theoretically thoroughly studied, and in industry often applied formalism for the modelling and verification of concurrent systems. CSP has been successfully applied in areas as varied as distributed databases, parallel algorithms, train control systems [3], fault-tolerant systems [2], and security protocols [15].

Fixing one syntax, CSP offers different semantical models, each of which is dedicated to special verification tasks. The traces model \mathcal{T} , e.g., covers *safety properties*. *Liveness properties* can be studied in more elaborate models. *Deadlock analysis*, e.g., is best carried out in the stable-failures model \mathcal{F} , the failures-divergences model \mathcal{N} allows for *livelock analysis*. The analysis of *fairness properties* requires models based on infinite traces, see [11, 1] for further details.

Recently, the CSP realm of models has been extended by the newly designed *stable-revivals model* \mathcal{R} [13]. On the practical side, the model \mathcal{R} is appropriate to study *responsiveness* [12], which is a significant property in the context of component-based system design. From a theoretical point of view, the model \mathcal{R} turns out to be fully abstract with respect to detecting when some system of processes can fail to make progress despite one or more of them having unfinished business with other(s). However, this comes at the price that certain algebraic properties fail to hold in the new model, among them the law \square - \square -distributivity $(P \square Q) \square R = (P \square R) \square (Q \square R)$ is the most prominent example.

CSP-Prover [5, 8, 6, 7] is an interactive theorem prover for CSP based on Isabelle/HOL [10]. With its theorem proving approach CSP-Prover complements the established model checker FDR [9] as a proof tool for CSP. CSP-Prover is generic in the various CSP models, currently, it fully supports the traces model \mathcal{T} and the stable-failures model \mathcal{F} .

CSP-Prover provides a deep encoding of CSP, and, consequently, also allows for meta theorems on the semantics implemented. Mistakes found, e.g., in the typing of the semantical functions of the predecessor of the model \mathcal{N} [17, 16], or in the algebraic laws for the model \mathcal{F} [6] demonstrates that, soon, presentations of similar models and axiom schemes will only be ‘complete’ once they have been accompanied by mechanised theorem proving [14].

In this paper, we report on an ongoing project of encoding the model \mathcal{R} in CSP-Prover, see Figure 1. CSP-Prover provides a large re-usable part, e.g., theories on CMS and CPO, which provide techniques for dealing with recursive process definitions, or the CSP syntax. On top of this re-usable part, each CSP

* This work has been supported by the EPSRC Project EP/D037212/1.

model needs to be defined individually with its domain, its semantical functions, and its proof infrastructure, where – thanks to the close relation between the CSP models – sharing and re-use is possible. Encoding a CSP model includes two major parts, namely *specifying the model* and *proving properties* about this specification. Due to Isabelle’s concept of conservative extensions, these two parts are often intertwined.

On the specification side, we first need to encode the domain of the model \mathcal{R} . Mathematically, this domain is a powerset reduced to ‘healthy’ elements. The notion of healthiness arises in a natural way, when the domain is interpreted as the collection of possible process observations. In Isabelle/HOL, we mirror this construction with a type definition. Already in this context, Isabelle/HOL requires the proof of properties, namely, that the newly defined types are non-empty.

The next step is to specify the refinement order, under which \mathcal{R} forms a complete lattice. Here, we declare our domain to be an instance of the Isabelle/HOL type class ‘order’. To this end we have to discharge the proof obligation that the inverse refinement order of \mathcal{R} is a partial order. We also prove that the inverse refinement order forms a pointed cpo: this result allows us to re-use CSP-Prover’s theory on CPO. As the model \mathcal{R} is currently only defined in the CPO approach, we refrain from linking our domain to CSP-Prover’s theory on CMS.

Next comes the specification of the semantical functions which map the CSP syntax to the domain. In Isabelle/HOL, these definitions come with proof obligations. Given, for example, healthy denotations of two processes P and Q , we have to prove that the denotation of the external choice ($P \square Q$) between P and Q is healthy as well. While Roscoe regards it in [13] as ‘mechanical calculation’ that all operators preserve \mathcal{R} ’s healthiness conditions, these proofs turn out to be a real challenge in Isabelle/HOL.

The final step in order to instantiate CSP-Prover with the model \mathcal{R} is to prove that all CSP operators are continuous w.r.t. the inverse refinement order. Given this continuity result, the semantics of recursive process definitions in CSP is defined via Tarski’s fixed point theorem, which also offers proof support in the form of fixed point induction.

The proof infrastructure for the model \mathcal{R} is based on the encoding described above. It consists of a collection of algebraic laws, which have been proven to be correct w.r.t. the encoding, as well as of a collection of tactics combining these laws.

Our instantiation of CSP-Prover with the model \mathcal{R} as laid out in this abstract is nearly complete: there are open continuity proofs, and more proof infrastructure needs to be developed.

References

1. A. E. Abdallah, C. B. Jones, and J. W. Sanders, editors. *Communicating Sequential Processes, the first 25 years*. LNCS 3525. Springer, 2005.
2. B. Buth, J. Peleska, and H. Shi. Combining methods for the livelock analysis of a fault-tolerant system. In *AMAST’98*, LNCS 1548. Springer, 1998.

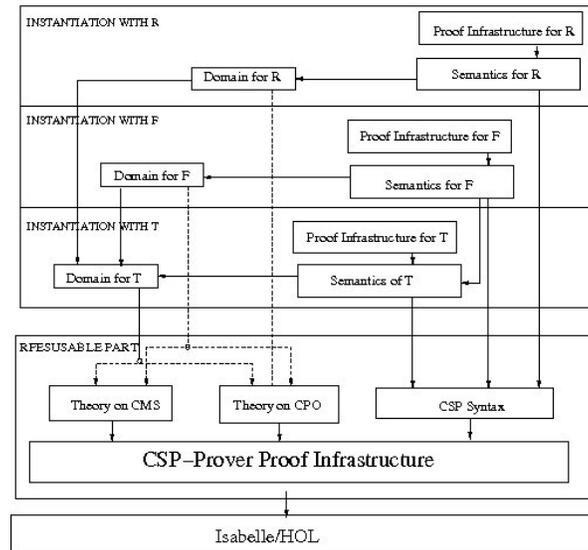


Fig. 1. Model R in CSP-Prover

3. B. Buth and M. Schröner. Model-checking the architectural design of a fail-safe communication system for railway interlocking systems. In *FM'99*, LNCS 1709. Springer, 1999.
4. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
5. Y. Isobe and M. Roggenbach. A generic theorem prover of CSP refinement. In *TACAS 2005*, LNCS 3440. Springer, 2005.
6. Y. Isobe and M. Roggenbach. A complete axiomatic semantics for the csp stable failures model. In *CONCUR 2006*, LNCS 4137. Springer, 2006.
7. Y. Isobe and M. Roggenbach. Proof principles of CSP – CSP-Prover in practice. In *LDIC 2007*. Springer, 2007. To appear.
8. Y. Isobe, M. Roggenbach, and S. Gruner. Extending CSP-Prover by deadlock-analysis: Towards the verification of systolic arrays. In *FOSE 2005*, Japanese Lecture Notes Series 31. Kindai-kagaku-sha, 2005.
9. F. S. E. Limited. Failures-divergence refinement: FDR2. <http://www.fsel.com/>.
10. T. Nipkow, L. C. Paulon, and M. Wenzel. *Isabelle/HOL*. LNCS 2283. Springer, 2002.
11. A. Roscoe. *The theory and practice of concurrency*. Prentice Hall, 1998.
12. A. W. Roscoe, J. N. Reed, and J. E. Sinclair. Machine-verifiable responsiveness. *AVOCS 2005*, 2005.
13. B. Roscoe. Revivals, stuckness and responsiveness, 2005. Unpublished Draft. Available at web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/105.pdf.
14. B. Roscoe, 2006. Private conversation.
15. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2001.
16. H. Tej. *HOL-CSP: Mechanised Formal Development of Concurrent Processes*. BISS Monograph Vol. 19. Logos Verlag Berlin, 2003.
17. H. Tej and B. Wolff. A corrected failure-divergence model for CSP in Isabelle/HOL. In *FME'97*, LNCS 1313. Springer, 1997.