Unifying Synchronous Systems

Dr. Andrew Butterfield
School of Computer Science & Statistics
Trinity College, Dublin 2

May 15, 2007

This document is an extract from research proposal CMSF186 submitted to Science Foundation Ireland for funding.

What is the question that this proposal addresses? This proposal will explore the development of a theory of hardware compilers within the UTP framework [HH98], thus bridging the synchronous-hardware, state-based, and message-based software domains under one theoretical umbrella. Furthermore, it will look at developing a parameterized collection of theories that allow straightforward verification of simple designs, whilst still being able to support reasoning involving the more complex properties required in more sophisticated systems. The key idea is to be able to integrate the simple and complex reasoning as required by the design of different parts of a system.

How will the question be addressed? A key observation in the work done on the operational semantics of Handel-C, and that work currently in progress for its denotational semantics, is that much of the complexity in the theory arises from the need to support relatively few language constructs, and even then only their more advanced features. In fact, most of the complexity can be laid at the feet of the so-called "`prialt`" construct, which allows a process to make a prioritised list of communication requests. Indeed, even for this construct alone, the full complexity is only required to support particular uses of one sub-construct, namely the "`default`" clause, which specifies the action to be taken if communication is not currently possible.

A typical response when providing a formal semantics for a real-world language, when faced with a few constructs with certain aspects that are complicated, is to simply rule them out for formal use, with a statement to the effect that if the language is to be used within a formal development framework, that certain language features are not supported. For Handel-C, the operational semantics published by the proposer [BW05] does not make any such restrictions. However, it does require a designer reasoning about a program without these features to use the full complex mathematical machinery, when for a specific problem instance, a much simpler and easier formal model might suffice.

A key feature of the UTP framework [HH98] is that it has a mechanism for linking different theories via a mathematical construct known as a *Galois connection*. This provides one of the key tools for unifying different frameworks, usually by showing how a system described in one framework can be viewed formally as a refinement of one developed in the other. For instance, work on a discrete-timed sequential process algebra can be linked formally to an un-timed sequential process algebra theory via a galois-link that captures how the timing information is lost when moving from the timed to the un-timed world [SH02].

This proposal is based on the observation that a denotational semantics for Handel-C corresponds to a discretely timed synchronous process algebra, and that there is a common theoretical framework independent of the fine detail and complexity needed to support Handel-C at all levels. The proposal is to develop a basic framework of discrete timed theories, which can then be parameterised by the amount of detail given about activity within a single discrete time-slot. Formal relationships between these levels of detail should the lead to corresponding Galois connections between the different theories, enabling them to be mixed and used together as appropriate. The entire discrete-timed theory framework will itself be linkable back to the un-timed process theory [WC02], thus allowing it to act as a high-level specification formalism for development of these systems. Another gain from the linkage between synchronous/timed theories and the un-timed variant will be the ability to handle complex systems built from multiple clock domains. The Handel-C language itself produces multiple-clock synchronous domains interconnected by asynchronous (un-timed) interfaces.

While the original work done focussed on the semantics of Handel-C, and the research proposal here takes its inspiration from that work, the intention is to develop a theoretical framework to support a broad class of similar hardware compilation systems. It is also intended that the theory will be accompanied by some methodology, in particular supported by the fact that such a framework will allow a designer to see immediately the cost in terms of ease of reasoning to be paid when using certain language constructs.

The Theory in Detail: The Unifying Theories of Programming (UTP) was expounded in a book of the same name [HH98], and is a framework designed to support multiple theories of different programming paradigms, and the linkages between these. These are all expressed using a language of alphabetised predicates, used to describe relations, typically between the before- and after-states of a computation. In [HH98], we are presented with theories of imperative programming languages, logic and functional languages, machine code, and reactive system languages like ACP and CSP. Also described are multiple theories of a single language capturing its semantics in different ways (axiomatic, denotational, operational) as well as linkages between these alternatives that allow their interrelationship to be explored. Also described is how features of one theory (e.g. assignment to state in an imperative language) can be incorporated into another theory (CSP).

A theory is written using *alphabetised predicates*, where the alphabet of a predicate covers all its free variables, and characterises the observations of interest regarding the computational systems being described. Typically there is a specification/programming language associated with the theory, and alphabetised predicates are used to give a semantics that language. As it is possible to write predicates which assert things that are physically/computationally infeasible, a number of *healthiness conditions* are introduced to constrain the predicates used to define the corresponding language constructs. The healthiness conditions are expressed as idempotent predicate-transformers, which when applied to a predicate return a healthy version, or leave it unchanged if already healthy. The foundational work for a theory consists of determining the appropriate observational variables and healthiness conditions, giving semantics to the language constructs, and validating algebraic laws relating terms in the language. This foundational work is often quite difficult, but its completion gives great insight into the nature of the paradigm under study. The utility of a theory comes from the resulting range of algebraic laws that can be used to reason about specifications or programs written using that language.

A key feature of UTP is that directly supports *refinement*, the process of converting a specification into an implementation. In UTP, refinement is simply logical entailment: a program P satisfies specification S if the predicate for P entails that for S, for any possible values of the observation variables. Like Z and CSP, UTP views the process of verifiably transforming specifications into corresponding implementations as the main rôle that its formalism is intended to support. Another important feature is that different theories can be linked together via "linking predicates", which under appropriate conditions form Galois connections between the two theories. These "Galois links" not only allow formal cross-analysis of linked theories, but also allow notions of refinement to bridge from one theory to another.

Since the publication of [HH98], there has been considerable interest in UTP [UTP06], and in particular work has been done on using it as a framework to build a formal theory called Circus that merges both Z and CSP, with support for refinement. The Circus project is led by Professor Jim Woodcock at the University of York, and has already had a number of successes in industrial applications [OCW04,OCW05,CCO05] A major focus of the effort is also in the area of developing tools to support Circus, ranging from parsers and type-checkers to model-checkers and theorem proving support. More recent work by others has looked at a timed-variant of Circus[SH02].

Motivation: The proposer has been collaborating with Prof. Woodcock, publishing mainly to date on the area of formal semantics for Handel-C [BW03,BW05,Cor05,BW06]. However, recent work, currently being prepared for publication, explores the idea of constructing a family of related UTP theories that talk about the introduction of time-slots at various levels of detail. It is this work that motivates the ideas behind this research proposal, which will marry the UTP ideas with the work done on Handel-C semantics.

By following this approach, we would expect to get theories that are very Circus-like, and which can be linked via a Galois connection back to Circus. This would increase the impact of this research as it would allow it to leverage off existing Circus results. Many of the applications of Circus are in areas where Handel-C is a candidate for implementing the systems concerned. In addition, it would allow us to exploit the current work on Circus tool-support, as much of this could be adapted to handle the new theories without anything like the amount of effort that would be require to build tools *ab initio*.

Methodology: The research can be viewed as a collection of workpackages: Foundations; Language; Refinement; Case-study; and Automation. We shall give a brief description of each.

Foundations are concerned with the key infrastructure of each theory, namely the choice of observations variables, and the required healthiness conditions. A key aspect of this workpackage will be the identification of the "Galois-links" between these theories.

Language While the target language, Handel-C, or similar, is already well-defined, it will be necessary to give it an appropriate formal semantics within the various theories developed as part of the foundations. The language constructs that only require the simplest of semantic models, will be given a semantics in all of the theories, where those in the more complex theories will be appropriate embeddings of the simpler theories. The Galois connections between these theories will define that embedding. Those language constructs of a more complex nature (like `prialt`, `default`) will only have semantics defined in the theories rich enough to handle them. Interestingly however, the Galois links will allow us to assess their effect in the simpler theories. This would allow us to answer questions regarding what can be proved about a program using complex constructs, if we limit ourselves to a simple theory. To make the theories useable, we then need to determine the relevant algebraic laws relating language constructs, and validate these against the underlying theories.

Refinement As the process of refining specifications down to code is important, we need to characterise the laws of refinement that capture the formal relationship between descriptions of a system at different levels of abstraction. Whilst the formal notion of refinement, logical entailment, is standard throughout UTP, we need to know the details of how such entailment links specification and program statements. This detail is necessary if we are to be able to effectively apply the theory in practice.

Case-Study The primary focus on this research is on establishing the foundations and laws associated with the corresponding theories. However, to ensure the theory is useful (i.e good), we need to concentrate on laws and properties that are actually needed when working with such a theory on a real problem. The point of doing case-studies is not simply to show that the theory is useful, but also to guide the emphasis and focus of the theoretical work, particulary at the earlier stages when there are a lot of choices to be made.

Automation Using formal techniques in practice involves having to do a large number of often very small, similar and not very interesting proofs, mixed with large complex involved and very detailed proofs. Both of these are very hard to do by hand, because of their dullness, and or the need for large amounts of book-keeping. It is generally recognised that industrial application of formal techniques now mandates the use of tool support. This proposal cannot realistically offer to develop both the theory and comprehensive tool support—such tool support could be a realistic goal of a subsequent proposal. However, it may be possible to get some limited progress here by performing minor adaptations to the tools currently being developed for Circus.

Collaboration: A key feature that led to the success of the proposer with the work done to date for Handel-C was the alternation of his own work periods with occasional collaborative research visits involving Professor Woodcock. It is intended that this proposal would continue this pattern of collaboration, as it has proved to be very effective.

Bibliography

[BW03] Butterfield, A. and Woodcock, J., "A "Hardware Compiler" Semantics for Handel-C", in Hurley, T. (Ed.), *Proceedings of the Second Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2002)*, Electronic Notes in Theoretical Computer Science, Elsevier Vol. 74, pp1–20, 2003.

[BW05] Butterfield, A. and Woodcock, J., "`prialt` in Handel-C: an operational semantics", in *International Journal on Software Tools for Technology Transfer*, Springer, Vol. 7, No. 3 pp184–203, June 2005.

[BW06] Butterfield, A. and Woodcock, J., "A "Hardware Compiler" Semantics for Handel-C", in Seda, A.K. *et al* (Eds.), *Proceedings of the Third Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2004)*, Electronic Notes in Theoretical Computer Science, Elsevier Vol. 161, pp73–90, 2006.

[CCO05] Cavalcanti A.L.C., Clayton P., and O'Halloran C., "Control Law Diagrams in Circus" in J. Fitzgerald, I. J. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods*, volume 3582 of Lecture Notes in Computer Science, pages 253 - 268. Springer-Verlag, 2005.

[Cor05] Corcoran B.J., *Testing Formal Semantics:Handel-C* , M.Sc (taught), University of Dublin, 2005.

[HH98] Hoare, C.A.R. and He, J., *Unifying Theories of Programming*, Prentice Hall, 1998.

[OCW04] Oliveira M.V.M., Cavalcanti A.L.C., and Woodcock J.C.P., "Refining industrial scale systems in circus", in Ian East, Jeremy Martin, Peter Welch, David Duce, and Mark Green, editors, *Communicating Process Architectures*, Concurrent Systems Engineering Series, vol. 62, pp281–309, IOS Press, September 2004.

[OCW05] Oliveira M.V.M., Cavalcanti A.L.C., and Woodcock J.C.P., "Formal development of industrial-scale systems." in *Innovations in Systems and Software Engineering*, 1(2):125–146, 2005.

[SH02] Sherif, A. and He, J., "Towards a Time Model for Circus" *Formal Methods and Software Engineering, 4th International Conference on Formal Engineering Methods, ICFEM 2002*, LNCS 2495, pp613–624, 2002.

[UTP06] Dunne S. and Stoddart B., editors, *Unifying Theories of Programming: First International Symposium, UTP 2006, Walworth Castle, County Durham, UK, February 5-7, 2006, Revised Selected Papers*, volume 4010 of Lecture Notes in Computer Science, Springer-Verlag, 2006.

[WC02] Woodcock, J. and Cavalcanti, A., "The Semantics of Circus", in *ZB 2002: Formal Specification and Development in Z and B*, LNCS 2272, pp184–203, 2002.