Formalising the Interface between Software and Hardware

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

April 22, 2008

This document is an extract from research proposal CMS1277 submitted to Science Foundation Ireland for funding.

What is the question that this proposal addresses? Verifying the correctness of safety-critical computer systems, particularly those that control machinery and vehicles, is an important but complex task. Many disparate aspects of the system need to modelled, each with their own peculiarities and interactions. This is why one of the Grand Challenges in Computer Science, GC6, on "Dependable Systems Evolution" [JOHW06] focusses on techniques to mathematically verify the correctness of such systems. The goal of GC6 is to develop a repository of verified software components, consisting of formal specifications and software implementations along with proofs of correctness supported by automated proof tools and proof assistants.

GC6 is a long-term research programme with a time horizon of ten to fifteen years. At present it is engaged in a series of research activities based around industrially inspired case-studies: recent such studies have included, for example, the validation of the Mondex smart-card's security protocols [SCW00]. A more recent case-study that has been adopted is that of a verified file-store for use in mission critical applications, suggested by NASA researchers who need such high-dependability systems for space research missions [JH05].

This proposal will build on the proposer's initial work in this area, namely the formal modelling of flash memory systems [BWa07], to use this case-study within GC6 to develop a comprehensive theoretical foundation for reasoning about hardware systems and their interface to software. This is viewed as being complementary to the notion of "hardware/software co-Design" [EWM01], that focusses on techniques for partitioning a system specification into hardware and software components in an efficient manner.

How will the question be addressed? The goal for this GC6 case-study is a collection of formal models of file-stores from the POSIX level down to their

implementation on hardware. Formal specifications of POSIX-like filesystem APIs exist [AZKR04,MS84] as do some of the refinements to lower algorithmic levels. The recent ICECCS paper [BWa07] starts to build a formal description of the flash memory technology used for filestores. Work is also commencing on building formal models of flash memory algorithms and data-structures as described in an ACM Survey paper by Gal and Toledo [GT05].

This research proposal seeks to explore the space between the hardware and the algorithms, by establishing a series of formal models at each level, linked by formal refinement steps. As the key concerns and nature of these models changes considerably moving between the hardware world (finite-state machines, clock-speeds, power-consumption, synchronisation, true parallelism) and the software world (data-structures, algorithms, execution complexity, interleaving concurrency), there is a pressing need for a way to link different theoretical paradigms. The complexity of these systems also requires the use of tool-support for modelling, hence its importance to GC6.

It is also clear that much work is still needed in order to understand the correctness and performance of these algorithms—as exemplified by the following abridged quote from the conclusions of [GT05]:

> "Unfortunately, many of these techniques are only described in patents, ... almost never contains a realistic assessment of the technique that it presents ... usually contains no quantitative comparisons to alternative techniques and no theoretical analysis. ... is almost always one-sided, describing the advantages of the new invention but not its potential disadvantages. ... often describe a prototype, not how the invention is used in actual products. ... are sometimes harder to read than articles in the technical and scientific literature."

What is clear is that there is considerable scope for formal and rigourous analysis of these algorithms, and for the development of new algorithms whose correctness has been verified and for which firm performance bounds are available.

A particular challenge, that becomes very evident when surveying literature on flash memory technologies [GT05], is the need for algorithms to perform wear-levelling, i.e to spread out memory writes and erasures because a memory-segment's lifetime depends on how often these operations occur. In order to reason about the correctness, (or rather the effectiveness) of such algorithms, we shall have to deal with notions of probability. There are two fairly obvious distinct ways in which probability interacts with possible wear-levelling algorithms. The first is that a wear-levelling algorithm may itself be deterministic, but we want to be able to establish that the probability of the algorithm failing as a whole is below some threshold, or that the useful lifetime of the system is (most likely) above some minimum threshold (time elapsed/no of operations). The second is that the performance of algorithms can be improved by having them make probabilistic choices [MR95]. A third reason to explore the interaction of wear-levelling and probability is that the state maintained by these algorithms, in order to track the extent to which memory segments have been

used, must itself be stored in some of those same memory segments, and hence is prone to such wear-related errors.

The need to explore the relationship between probability and algorithms is by no means restricted to the flash memory technology that we are proposing as our main case-study. The techniques we plan to develop would support formal reasoning about other application areas, which we can classify into two broad camps: randomised algorithms, and fault-tolerance in the face of erroneous component behaviour.

Randomised algorithms [MR95,MM04,Sub07] use randomness to make choices, typically where there is a notion of "best choice" but this is expensive to determine. Often such algorithms obtain near-optimal solutions far more efficiently than deterministic ones that search for good choices. Application areas are very diverse covering problems such as choice coordination [MM04], binary constraint solving [Sub07] and symmetry breaking [HS06]. Theoretical results in this area include the notion of "almost-certain" termination [HHo07] showing how to establish that an algorithm with randomized choices does progress over time towards completion.

In the fault-tolerance area, the problem is to ensure failure-free behaviour of a system even when sub-components may develop faults unpredictably. Typically it is impossible to make such a system totally failure-free, and so the main concern is showing that the probability of failure is below some acceptable threshold [CGK07,MSm07].

We should stress at this point that the techniques we plan to develop provide a range of semantic models for what will be an integrated collection of specification, software and hardware description languages. As such, they will have an applicability and generality far beyond the scope of the case study around which this proposal is focussed.

This proposal seeks to explore the use of the "Unifying Theories of Programming" (UTP) framework [HH98] to address this challenge. A key feature of the UTP framework is that it has a mechanism for linking different theories via a mathematical construct known as a *Galois connection*. A Galois connection links two ordered mathematical structures, often with different levels of detail, in a coherent and order-consistent manner. 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. This proposal would build on emerging results involving probability within UTP [HS06] in order allow reasoning. In terms of tool support, we expect to explore techniques such as SMT solvers [BP07,CGK07] as a means for automating the models and deductions required.

In [HS06], Galois connections are used to capture the notion of refinement between two semantic models (and their corresponding sets of laws). The first model, used for specification, has no notion of probability, but does have a notion of non-deterministic choice. The second model extends the first by adding in a probabilistic choice operator. This work is re-visited in [He07], where a

Galois connection is used to show how the semantics of a given language can be extended with new features. The key point is that the Galois connection facilitates the calculation of the extended semantics functions, rather than requiring an invent-and-verify approach to determining the extended semantic model. One of the extensions discussed is probability, and it is worth pointing out that both [He07] and [HS06] discuss a model where any system with a non-zero probability of failure is deemed to be indistinguishable from certain failure (the "possible failure as failure" model). What is of more interest in fault tolerant systems is the interpretation of a small failure probability as success (the "unlikely failure as success" model), as exemplified by the "almost-certain termination" approach addressed in [HHo07].

The ideas just presented are very large indeed, and this proposal will focus on a smaller but key part, The goal will be to use this case study to take formal modelling and refinement theory (as embodied in UTP) to the point where it can formally describe flash memory (both functional and probabilistic aspects) the intermediate hardware level and the lower software layers, using appropriate Galois links to propagate probabilities up to the software layer, and to ensure the correctness of refinements down to the hardware level. The Gal and Toledo paper [GT05], as well as the Flash Translation Layer (FTL) standard published by Intel [Int98] will serve as guidelines to focus the case study.

The Theory in Detail: We now present a full, but highly simplified example, based on [HS06], to make clear what the proposed technique entails. We consider a simple programming language with two atomic actions, skip and assignment, and two compound actions, sequential composition and non-deterministic choice. The skip statements ($Skip$) simply does nothing, whilst assignment $x := e$ updates variable x to take on the value of expression e. Sequential composition joins two statements together ($A; B$), and executes A first, and starts B once (if!) A has terminated. Non-deterministic choice $A \sqcap B$ simply executes either A or B according some unknown choice criteria — this may seem a strange construct in a programming language, but is very useful in specifications to avoid providing too much detail too early. The process of refinement from specification to code can be viewed semantically as a process of reducing the amount of non-determinism present. We can give this small language a partial-correctness semantics in UTP by considering the key observation to be that of the program state s, a mapping from variable names to their current values. We then write a predicate that relates the state before a program is executed (s) to that after the program has terminated (s'). So for example, the $Skip$ statement has the semantics $s' = s$, i.e. the state is unchanged. The assignment statement ($x := e$) semantics states that s' is simply s with x updated to its new value. The semantics of sequential composition ($A; B$) asserts the existence of a mid-state s_0 such that A transforms s to s_0, while B converts s_0 into s'. The outcome of non-determinism ($A \sqcap B$) is any

outcome that satisfies the semantics of either A or B.

$Skip$	$s' = s$
$x := e$	$s' = s \oplus \{x \mapsto e_s\}$
$A; B$	$\exists s_0 \bullet A[s_0/s'] \wedge B[s_0/s]$
$A \sqcap B$	$A \vee B$

We now consider developing a probabilistic view of the language semantics, so we can extend it with a probabilistic choice operator $A \,{}_p\!\oplus\, B$, which can choose to execute A with probability p, and B with probability $1 - p$. In order to do this we need a richer semantic model, one that assigns a probability to each possible program state. Our key observation is now a probability function pr over every possible program state. We now define the semantics of each language construct as a relation between the initial probability distribution (pr) and the final one (pr'). The semantics of $Skip$, ; and $\sqcap$ change very little (at least in how they are written), but assignment becomes more complex: The probability of a given after-state s', (written $pr'(s')$) is now the sum of $pr(s)$ taken over all before-states s that are transformed into s' by the assignment. The semantics of the probabilistic choice simply computes the appropriately weighted sum of the probabilities of the two alternatives.

$x := e$	$pr'(s') = \sum_{\{s \mid s' = s \oplus \{x \mapsto e_s\}\}} pr(s)$
$A \,{}_p\!\oplus\, B$	$\exists pr_A, pr_B \bullet A[pr_A/pr'] \wedge B[pr_B/pr'] \wedge pr'(s') = p \cdot pr_A(s') + (1-p) \cdot pr_B(s')$

The key point is that we can relate the two theories using a Galois connection. The idea is that we provide a linking predicate that explains how to convert between the model that observes state (s, s') and that which observes state probabilities (pr, pr'). For example, we may decide that any behaviour with non-zero probability is possible and so suggest the following linking predicate: $pr(s) > 0 \wedge pr'(s') > 0$. Given such a linking predicate, a standard technique [HH98,pp40–41] gives us a logical Galois connection linking the two theories. This connection allows to take a specification with non-determinism in the partial-correctness model, and move it into the probability domain, with the subsequent replacement of non-determinism by probabilistic choice as a refinement.

The above example is very simplified as we need to deal with far more features in practice: total correctness, iteration, recursion, concurrency, communication and time. This means that both semantic domains need more observation variables, and the linkages become more complex. Also the choice of linking predicate is not set in stone —different choices here give different linkages between the theories, and work is needed to establish what works best for any given application concern.

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

brief description of each, and then discuss how they would be scheduled and organised.

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 determination of algebraic laws for the semantics.

Refinement We need to characterise the laws of refinement that capture the formal relationship between descriptions of a system at different levels of abstraction. The design and choice of these Galois-links is the key to finding useful and tractable refinement techniques.

Case-Study This research proposal is case-study driven, taking flash memory filestores as an application area for two primary reasons: (i) the case-study chosen allows this research to link into the emerging CG6 research community, thus increasing both its relevance and impact; and (ii) the use of a "real-world" case study like this ensures that the theoretical work is kept aligned with the kinds of problems that system developers need to address, rather than simply those which admit a nice mathematical formulation.

Automation It is generally recognised that industrial application of formal techniques now mandates the use of tool support. This proposal will explore the use of tool support largely by exploring how its ideas can be encoded in existing theorem proving and model-checking tools.

Collaboration: A key feature that lead to the success of the proposer with the work done to date for ONFI and flash memory was the alternation of his own work periods with occasional collaborative research visits involving Professor Woodcock. The purpose of the visits was to discuss and resolve issues and problems, seeking insight from our differing perspectives and expertise, as well as planning suitable avenues of mutual interest for future work. These visits proved very useful in keeping the research work moving, and at the cutting edge. It is intended that this proposal would continue this pattern of collaboration, as it has proved to be very effective.

In addition to this specific collaboration, attendance at appropriate workshops and conferences will help to move the work along. In particular, participation in the GC6 community meetings and their tri-annual conference (Verified Software: Theories, Tools, Experiments (VSTTE), first held in Zurich 2005, next due in Toronto, late 2008), will prove to be very valuable.

Bibliography

[AZKR04] Arkoudas K., Karen Zee K., Kuncak V. and Martin C. Rinard M.C., "Verifying a File System Implementation", *Formal Methods and Software Engineering, 6th International Conference on Formal Engineering Methods, ICFEM 2004, Seattle, WA, USA, November 8-12, 2004, Proceedings*, Springer, LNCS 3308, pp373–390, 2004.

[BP07] Brown G.M. and Pike L., "Temporal Refinement Using SMT and Model Checking with an Application to Physical-Layer Protocols" *5th IEEE/ACM International Conference on Formal Methods and Models for Codesign, 2007, MEMOCODE 2007,* May 30th–2nd June 2007, Nice, France, pp171-180, IEEE Press 2007

[BSW07] Butterfield A., Sherif, A., and J. Woodcock, J., "Slotted Circus: a UTP-Family of Reactive Theories', in *Integrated Formal Methods, 6th International Conference, IFM 2007, Oxford, UK, July 2007, Proceedings*, Springer, LNCS 4591, pp75–97, 2007.

[But07] Butterfield A., "A Denotational Semantics for Handel-C", in *Formal Methods and Hybrid Real-Time Systems*, Springer, LNCS 4700, pp45–66, September 2007.

[BWa07] Butterfield A., and J. Woodcock, J., "Formalising flash memory: first steps", in *ICECCS: 12th IEEE International Conference on Engineering of Complex Computer Systems 2007*, Auckland, New Zealand, 2007, IEEE Computer Press, 2007.

[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.

[CGK07] Colvin, R., Grunske L. and Winter, K., "Probabilistic Timed Behaviour Trees", in *Integrated Formal Methods, 6th Intl. Conference, IFM 2007*, Springer, LNCS 4591, pp157–175, July 2007.

[DB06] Dowse, M. and Butterfield A., "Modelling Deterministic Concurrent I/O", in *11th ACM SIGPLAN International Conference on Functional Programming (ICFP 2006), Portland Oregon*, pp148–159, Sep 2006.

[Dow06] Dowse, M., *A Semantic Framework for Deterministic Functional Input/Output* Ph.D. thesis, University of Dublin, 2006.

[EWM01] Ernst R., Wolf W. and de Micheli G. (eds.), *Readings in Hardware/Software Co-Design*, ISBN 1558607021, Morgan Kaufmann, 2001.

[GT05] Gal E. and Sivan Toledo S., "Algorithms and data structures for flash memories", *ACM Comput. Surv.*, Vol. 37, No. 2, pp138–163, ACM Press 2005.

[He07] He J., "Linking Semantic Models", *Theoretical Aspects of Computing—ICTAC 2007*, Springer, LNCS 4711, pp18–33, September, 2007.

[HHo07] Hallerstede S. and Hoang T.S., "Qualitative Probabilistic Modelling in Event-B*", in *Integrated Formal Methods, 6th Intl. Conference, IFM 2007*, Springer, LNCS 4591, pp293–312, July 2007.

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

[HS06] He J. and Sanders J.W., "Unifying Probability", *Unifying Theories of Programming, First International Symposium, UTP 2006, Walworth Castle, County Durham, UK, February 5-7, 2006, Revised Selected Papers*, Springer, Lecture Notes in Computer Science, 4010, pp173–199 2006.

[Int98] Intel Corporation, "Understanding the flash translation layer (FTL) specification", Application Note 648, Intel Corporation.

[JH05] Joshi, R. and Holzmann, G.J., "A Mini Challenge: Build a Verifiable Filesystem", *Formal Aspects of Computing* Springer Vol 19. No. 2, pp269–272, June 2007

[JOHW06] Jones, C., O'Hearn, P. and Woodcock, J., "Verified Software: A Grand Challenge", *IEEE Computer*, Vol. 39, No. 4, pp93–95, April 2006.

[MM04] McIver, A. and Morgan C., "Developing and Reasoning about Probabilistic Programs in *pGCL*", *Refinement Techniques in Software Engineering*, Springer, LNCS 3167, pp123–155, November 2004.

[MR95] Motwani R. and Raghavan P, *Randomized Algorithms*, Cambridge University Press, 1995.

[MS84] Morgan C. and Bernard Sufrin B., "Specification of the UNIX Filing System", *IEEE Transactions on Software Engineering*", Vol. 10, No. 2, pp128–142, March 1984

[MSm07] Meinicke L. and Smith G., "A Stepwise Development Process for Reasoning About the Reliability of Real-Time Systems", in *Integrated Formal Methods, 6th Intl. Conference, IFM 2007*, Springer, LNCS 4591, pp439–458, July 2007.

[SCW00] Stepney, S., Cooper, D. and Woodcock, J., "An Electronic Purse: Specification, Refinement, and Proof", *Technical monograph, Oxford University Computing Laboratory*, PRG-126, July 2000.

[Sub07] Subramani K., "A Randomized Algorithmfor BBCSPs in the Prover-Verifier Model", *Theoretical Aspects of Computing—ICTAC 2007*, Springer, LNCS 4711, pp455–466, September, 2007.

[TMBH06] Tyrrell, M., Morris J.M., Butterfield A. and Hughes, A., "A Lattice-Theoretic Model for an Algebra of Communicating Sequential Processes", in *3rd Intl. Colloq. on Theoretical Aspects of Computing (ICTAC06)* Nov. 2006.

[Tyr04] Tyrrell, M., *Towards a Formal Method for Distributed Object-Oriented Systems* Ph.D. thesis, University of Dublin, 2004.