

# Research Manifesto for Activities Related to WG 2.11 — Generative Programming

Kevin Hammond\*

April 30, 2004

## 1 Current and Near-Term Research Activities

I lead the Functional Programming research group at St Andrews, with 6 members at the moment. My current research activities fall into two main categories, both relevant to WG 2.11:

1. Hume: a domain-specific programming language targetting real-time embedded systems;
2. parallel and Grid computing based on Haskell.

### 1.1 Hume

The primary purpose of the Hume language design is to explore the expressibility/costability spectrum in resource constrained systems such as real-time/embedded systems. A subgoal is to explore the feasibility of applying functional programming technology to hard real-time, hard space situations, so overcoming a perceived barrier to the use of this technology.

Hume provides a number of very high level features including higher-order functions, polymorphic types, arbitrary but sized user-defined data structures, asynchronous processes, lightweight exception handling, automatic memory management and domain-specific metaprogramming features, whilst guaranteeing strong space/time behaviour. Hume is unusual in being based on a novel combination of finite state machines with recursive function definitions rather than the more usual propositional logic, or flat finite-state-machine models, and in being structured as a series of levels, each of which exposes different machine properties.

We have produced determinate cost models for Hume levels from HW-Hume (a simple language for hardware design) up to PR-Hume (incorporating primitive recursive functions and inductive data structures). At present, the cost

---

\*School of Computer Science, University of St Andrews, St Andrews, Scotland. Email: [kh@dcs.st-and.ac.uk](mailto:kh@dcs.st-and.ac.uk).

models are restricted to stack and heap consumption, and have been verified against actual prototype machine implementations. Our research is now proceeding in a number of directions (for which we have applied for €1.5M funding in collaboration with LMU, München, Germany; AbsInt GmbH, Saarbrücken, Germany; LASMEA, Clermont-Ferrand, France; and Heriot-Watt University, Scotland; and also for UK research council and defence funding).

These research directions are:

1. constructing formal proofs of the correctness of our analyses;
2. improving the quality of automatic memory management;
3. adapting Hume to real embedded systems applications, including the Cycab autonomous vehicle;
4. developing distributed Hume systems and applications;
5. studying how to write more robust device driver code using Hume;
6. developing accurate time modelling based on abstract interpretation and good hardware models;
7. exploiting staged metaprogramming for costing Hume levels, so allowing costable transitions between Hume levels.

We envisage a possible connection with the EC initiative on Global Computing, and are pursuing this for future funding. We are also pursuing connections with Symbian, British Aerospace and Intel.

## 1.2 Parallel and Grid Computing

The notion of a *skeleton* or *pattern* capturing computational structure has now become widespread. One common application is to parallel computing, where such patterns can reduce the cost and complexity of producing parallel code, by allowing algorithms to be specified as a conjunction of common patterns of parallel computation. *Implementation skeletons* can be produced, corresponding to implementation-specific instances of high-level skeletons. Such implementation skeletons must, however, be selected by the programmer. We have studied the problem of selecting appropriate implementations *automatically at compile-time* using Template Haskell meta-programming constructs driven by static cost models. The meta-programming approach has the advantage of eliminating dynamic overheads caused by introducing *adaptors* to match high-level specifications to specific implementation skeletons, while still allowing a single source program to be targeted to multiple platforms and architectures. Moreover, implementation skeleton parameters can be automatically tuned at compile-time to suit a target architecture. Whilst still at an early stage of development, our results suggest that this should provide a flexible, and hopefully effective, means to write efficient parallel programs.

The idea can be extended to Grid computing, where there are major problems in managing scheduling systems. We envisage *autonomic*, self-adapting, systems capable of responding to dynamic changes in execution patterns, and of predicting future changes based on past, present and profiling information. It may be possible to apply generative programming technology either to construct scheduling templates for Grid applications or to construct specific schedules for certain applications. This work is at a very early stage, however.

We are constructing an EU research proposal to pursue this possibility in collaboration with the University of Marburg, Germany; and various companies including SAP, IBM, Intel etc.

## 2 Future Activities

In order to cement WG2.11, I suggest investigating an EU Coordination Action (CA). CAs are relatively lightweight mechanisms, aimed at bringing together experts on a regular basis to develop a technical research agenda. While WG2.11 serves a similar purpose, a CA has two main advantages:

1. it provides funding for additional outreach actions, which may increase the effectiveness of the working group;
2. it provides a focus for talking to industry;
3. it provides a route to influence the European Commission's research agenda, both directly and indirectly — coordination actions often lead to new research programmes.

I would be happy to help with (or perhaps coordinate) such a proposal. I have some experience with the Framework VI review process, which should help.