

## Research Plans in Program Generation:

# Dynamic cross-component performance optimisation

Olav Beckmann and Paul H J Kelly  
Software Performance Optimisation Group, Imperial College London  
(beckmann@imperial.ac.uk, p.kelly@imperial.ac.uk)

Software systems are being built from increasingly complex component libraries. Much of this complexity comes from variants and parameters used to optimise a component to its context of use. Apart from requiring heroic efforts from library implementors, this approach also suffers from crucial problems in terms of performance and usability. Instead, future software systems should be built from light-weight component libraries which are composed by powerful domain-specific optimisation components (DSOCs). We are exploring this idea using automatic program generation, combined with iterative optimisation, in order to adapt light-weight software components (a) with respect to the platform they are executing on, (b) with respect to runtime parameters and (c) with respect to dynamic calling context.

**Theoretical Foundation for Multi-Stage Generative Optimisation Components.** The long-term objective of our research is to develop a theoretical framework for multi-stage, iterative optimisation of software components.

- To develop multi-stage component summary metadata that describes optimisation constraints, context and performance information, and is enhanced as run-time information is used.
- To develop metadata for optimisation components that describes the code transformations the optimisation components perform, as well as characterising the mapping of component metadata from source- to destination-variants.
- To characterise the cost of multi-stage optimisations, as a function of the context we are optimising for and of the available previous optimisations.
- To develop a model of the performance benefits of optimisations, supported by runtime performance measurement when components are used.
- To characterise the compositional properties of optimisations: for example, given that we have found an optimal tiling, can we do unrolling next without re-doing the tiling?
- To develop algorithms for deciding when multi-staged optimisation should be performed. The scenario is that in a running system, a component is being used in a new context (which could simply be the information, based on runtime profiling, that this component is frequently used). The decision that has to be taken is whether to re-optimize, which previous optimisations to re-use and which optimisations to apply now.
- To develop a system for automatically caching the results of re-usable optimisations and for selecting the optimal precursor code when re-optimisation is performed.

**Infrastructure.** It is an interesting aspect of research into domain-specific program optimisation that the core research issues are concerned with domain-generic aspects. We are planning to carry out further work on enhancing our prototype C++ metaprogramming tool, the TaskGraph library [1], and make a public release. Specific issues include type safety, multistage programming, and extending the metaprogramming facilities to make it easy to build domain-specific analyses and optimisations.

**Related activities.** Other projects in the Software Performance Optimisation group involve using these ideas in distributed Java, .Net and FPGA, Python and scientific visualization contexts.

## References

- [1] Olav Beckmann, Alastair Houghton, Paul H J Kelly, and Michael Mellor. Run-time code generation in C++ as a foundation for domain-specific optimisation. In *Domain-Specific Program Generation*. Springer Verlag, March 2004. LNCS 3016.