DEPARTMENT OF COMPUTER SCIENCE

THE UNIVERSITY of York

Programming Languages and Systems Research Group

How can the expressive power of different specification and programming styles be used most effectively? What are the limitations in combining them? How can multi–language technology be implemented most efficiently?

What parallel programming styles are truly architecturally–independent? What techniques enable the matching of algorithmic structures to machine structures?

How can the effectiveness of software specification and verification methods be improved? Can structured and formal techniques be integrated?

How can Formal Methods best assist in the task of system design and verification? How are design decisions constrained by the process of formal refinement? What tool support is needed to make these methods a practical reality?

What is a good notation for exploring design choices in algorithm development? How can we reason about algorithms such as cryptographic protocols that can fail for quite subtle reasons?

This research group aims to advance programming languages, methods and tools, with a special interest in radical alternatives. We address both the principles underlying new kinds of languages and technologies for system's design and programming, and the practice of implementing and using them effectively.

Functional Programming work focuses on novel implementation techniques and programming tools. We have developed *profiling tools* widely used to miniaturise lazy computations, and have a growing repertoire of novel *tracing methods* based on redex trails. We have designed and implemented both *combinator libraries* and *language extensions* that provide increased expressive power for specific application domains. We also continue to develop and distribute a *Haskell compiler*.

Parallel Computation will be the *only* way to solve problems faster when serial processors will eventually hit fundamental physical limits to their speed. We focus on *architecturally–independent techniques* for exploiting current and future parallel architectures. Much of our work uses a *coordination model* of concurrency and concentrates on building implementations on parallel machines, on modifying the model as theoretical work uncovers deficiencies and on applying it to a variety of application areas such as *image processing* and *knowledge–based systems*.

Software Specification, particularly for non–critical applications, is a low–quality task in many situations. Our work on integrating *formal specification* within existing *diagram–based analysis and design*, is at the centre of work to improve the specification and development of software. Research in the specification language Z work focuses on *patterns* and on *translation templates*. In addition, we work on an *information systems* variant of the UML specified within the B method.

Reactive–Systems Design is a key task in embedded–systems development. We work on improving design technologies and tools by strengthening their mathematical foundations. Our focus is on developing methodology– and tool–friendly semantics of *engineering design languages* (e.g., Statecharts and Esterel), on promoting *multi–language paradigms* (mixing operational and declarative styles), and on devising novel *model checkers* for validating asynchronous systems designs.

Graph Transformation combines the strength of graphs in visualisation with a rule–based way of computing. We are developing the *graph programming language* GP which is based on a complete and minimal core language. Another topic is the *safety of pointer programs* in imperative languages: we have developed a static check for the invariance of pointer–data structures. Yet another topic is *term graph rewriting* which improves conventional term rewriting by sharing common subexpressions.

Reasoning about Algorithms, such as *cryptographic protocols*, is necessary to guarantee confidential and authenticated communication over a public network. Many proposed protocols have been found to contain deep and subtle flaws, and several formalisms proposed for reasoning about such protocols are likewise flawed. We focus on the question on how to make *formal proofs* easier to conduct, by enhancing *tool support* and devising novel *proof tactics*.

Functional Programming

Concurrent Programming

Coordination Models

Rigorous Information Systems

Reactive–Systems Design

Mathematics of Programming

Concurrency Theory

Formal Software Development

Formal Verification Technology

Graph Transformation and Rewrite Systems

Language Implementation

Programming and Validation Tools

Research Activities

Projects with current or recent external funding include: Advanced Redex Trails, Safe Pointers by Graph Transformation, Investigation of Cryptographic Protocols, Practical Formal Development and Rigorous Information Systems.

Our current or recent industrial collaborators include Logica and Microsoft. We collaborate with academics in Amsterdam, Augsburg, Bamberg, Berlin, Brisbane, Eugene, Gothenburg, Kiel, New York, Oldenburg, Paris, Williamsburg and Yale. We are represented on two IFIP Working Groups and are regularly involved in international workshops and conferences.

At York the research group does joint work with the Advanced Architectures and High–Integrity Systems research groups, as well as with research teams in security and quantum computation.

Academic and Research Staff and Students

Nuno Amálio, Research Student. *Formal software engineering.*

Néstor Cataño, Research Associate. *Verification and validation of Java programs, model checking.*

Mike Dodds, Research Student. *Graphs, pointer checking.*

Jonathan Ezekiel, Research Student. *Model checking.*

Xiaocheng Ge, Research Student. *Secure databases.*

Guanhua He, Research Student. *Machine supported reasoning about functional programming.*

Jeremy Jacob, Lecturer. *Formal methods for software development.*

Stephane Konstantaropoulos, Research Student. *Open source software, C language.*

Gerald Luettgen, Senior Lecturer.*Reactive-systems design, concurrency theory, model checking.*

Greg Manning, Research Student. *Graphs, graph–based programming languages and tools.*

Neil Mitchell, Research Student. *Pointer programs.*

Jan Tobias Muehlberg, Research Student. *Model checking intermediate languages and object code.*

Matthew Naylor, Research Student. *Functional programming, program transformation, hardware design and compilation.*

Detlef Plump, Senior Lecturer. *Theory and applications of graph transformation, graph–based programming models.*

Fiona Polack, Senior Lecturer. *Software development processes, formal and informal system models.*

Colin Runciman, Professor. *Functional programming, software technology, mathematics of programming.*

Thomas Shackell, Research Student. *Implementing functional programming languages.*

Sandra Steinert, Research Student. *Theory and application of graph transformation, graph–based programming languages and tools.*

Nur Izura Udzir, Research Student. *Capability–based coordination.*

Malcolm Wallace, Research Associate. *Functional programming, miniaturisation, tracing, testing, modelling.*

Andrew Wilkinson, Research Student. *Types and coordination.*

Alan Wood, Senior Lecturer. *Coordination technology, concurrency, parallelism.*

Further Information

For further information please consult the research group's web pages located at *www.cs.york.ac.uk/plasma*, email *plasma-request@cs.york.ac.uk*, or contact Prof. Colin Runciman at *+44 (0)1904 432740*. Individual group members can be reached by sending email to ⟨*Forename*⟩.⟨*Surname*⟩*@cs.york.ac.uk*.

Oct. 2005