IST-2001-33149 (MRG)

Periodic progress report: year 2

23rd February 2004

Project start date: 1 Jan 2002
Project duration: 36 months
Project coordinator: University of Edinburgh
Project partners: University of Edinburgh, Ludwig-Maximilians-Universität München

Funded by the European Community's "Information Society Technologies" Programme (1998–2002) under the FET proactive initiative on Global Computing.

Contents

1 Executive Summary

The MRG project started on 1st Jan 2002. The aim of the project is to develop the infrastructure needed to endow mobile code with independently verifiable certificates describing its resource behaviour. These certificates will be condensed and formalised mathematical proofs of a resource-related property which are by their very nature self-evident and unforgeable.

The MRG workplan is structured into ten technical workpackages (and one administrative workpackage) of which seven were scheduled for activity during year 2 with planned work including many core tasks. On top of this, the project was behind schedule at the end of year 1 due partly to delays in recruiting staff and partly to the fact that some key tasks turned out to be much more time-consuming than expected. Year 2 has therefore been a busy period for the project with effort focussed on core tasks at the expense of some tasks that are not on the critical path. We took advantage of underspend on personnel in Edinburgh and the availability of well-qualified researchers to recruit additional manpower at that site, which eased the situation considerably. The project is now much closer to being on schedule and most of the components of the planned infrastructure are now in place.

The deliverables for year 2 are in the form of reports or software prototypes. All reports and software are downloadable from the project website. The content of the deliverables and key decisions are explained in Section 2.2 below, and deliverable D11h compares progress with checkpoints and quality measures previously specified in the Self Evaluation Plan (D11f). Deliverables produced between 31st Dec 2003 and the date that this report was prepared are also mentioned, in order to accurately reflect what we will be prepared to present during the project review meeting. However, the work done on these since 31st Dec 2003 is not reflected in the figures on manpower.

Section 2.6 proposes some changes to the workplan for year 3, to eliminate some tasks that are peripheral to the core objectives of the project and add tasks that did not appear in the original workplan but now seem to be worthy of effort. This is in accordance with recommendations from the reviewers at the end of year 1.

2 Work progress overview

2.1 Specific objectives for the reporting period

As discussed during the first review meeting, our main goal during year 2 was to build on the solid infrastructure established during year 1 to provide the basic elements required for a full implementation of proof-carrying code for heap space bounds. Specifically, we hoped to:

- Complete the bytecode logic (task 2b) and its implementation (task 2c) and subsequent tasks in WP2 that aim to ensure the quality of the logic and its implementation;
- Implement a resource type system for heap space bounds for Camelot (tasks 4b and 4c); and
- Implement a function taking a Camelot program with a heap space bound described by a resource typing to a proof in the bytecode logic that the Grail code generated by the Camelot compiler satisfies that bound (see task 6a).

Additional priorities were tasks from WP3 (3d and 3e) that were scheduled for year 1 but had not been completed, and WP4 tasks from year 2 (4e and 4f). Secondary priorities were tasks in WP5 and WP10 that were scheduled for years 1 and 2.

Section 2.2 and the self-assessment report for year 2 (D11h) give an overview of the work done towards these objectives.

2.2 Overview of the progress of the project during the reporting period

The following table lists the deliverables that have been completed since the last report.

Del. no.	Deliverable name	WP no.	Est. person-months	Del. type	Planned delivery (month)	Complete?
D2a	Language of assertions for bytecode logic	2	2	report	5/02	9/03
D2b	Proof rules for bytecode logic	2	3	report	7/02	9/03
D2c	Proof checker for bytecode logic	2	4	prototype	4/03	11/03
D2e	Theorem prover for bytecode logic	2	3	prototype	9/03	11/03
D2f	Encoding of VM semantics in theorem prover [optional task]	2	1	prototype	5/03	5/03
D3d	Extend compiler with immutable objects and higher-order functions	3	4	prototype	1/03	11/03
D3e	Extend compiler with optimisations	3	3	prototype	11/02	9/03
D3f	Extend with mutable state and concurrency [optional task]	3	1	prototype	4/04	2/04
D4b	Type system for space-like resources	4	4	report	12/02	11/03
D4c	Typechecker for compiler	4	4	prototype	3/03	11/03
D4e	Proof of soundness over cost model	4	6	report	12/03	no
D4f	Proof of soundness over bytecode logic	4	4	report	8/03	no
D5a	Type system for expressing limits on parameter values	5	4	report	12/02	2/04
D5b	Proof of soundness for parameter value constraints	5	3	report	10/04	2/04
D5d	Resource type system for bytecode	5	6	report	6/03	no
D6a	Certificate generator	6	6	prototype	9/03	no
D7a	Extension of type system by allowing user annotations	7	3	report	11/03	no
D7b	Adaptation of bound generation/certification to optimisations	7	5	prototype, report	9/03	no
D10a	Practical effectiveness of mobile virtual machines	10	4	report	10/02	3/03
D10b	Metalanguage for describing virtual machine configuration	10	6	report	12/03	2/04
D11d	Workshop at end of year 2	11	0	workshop	12/03	3/04
D11h	Assessment of progress in year 2	11	0	report	1/04	2/04

Workpackages 2 and 3 are thereby completed.

Below is a verbatim copy of relevant parts of the MRG workplan, augmented with a description of what work has been carried out under each workpackage and task and explaining significant departures from what was foreseen. Workpackages and tasks on which work was neither planned nor carried out during year 2 are not included.

WP2: Definition of bytecode logic

Objectives: Development of bytecode logic, including language of assertions and proof rules, and a proof checker.

The purpose of this workpackage is to provide a language (2a) for making assertions about the resource usage of bytecode programs with respect to the cost model defined in 1b, and a logic (2b) for proving such assertions. The certificates that will be attached to dowloadable bytecode will be proofs in this logic (but see 9). Certificates must be easily checkable by the recipient, and the implementation of the proof checker (2c) will be part of the trusted code base (TCB); thus the logic and its implementation must be simple and generally acceptable. To ensure the quality of the logic and its implementation, we have included related tasks (2e and 2f) whose main underlying aim is assessment of these components, and do not directly feed into later work.

a. Develop **language of assertions**. *(2 months)*
Deliverables: technical report *Prerequisites: 1b*
We will develop a logical language for asserting resource-related properties of bytecode programs. This will probably take the form of a Hoare-style logic, with assertions making explicit reference to code fragments. The expressiveness and convenience of the notation will be checked using examples from 1b and its semantics will be formally defined.

Work carried out: This was submitted in year 1 but was not accepted by the reviewers. The original purpose of this task was as a precursor to 2b, so the results have been incorporated into a combined deliverable D2a/b.

b. Develop **proof rules**. *(3 months)*
Deliverables: conference/journal paper *Prerequisites: 2a*
We will develop rules for proving assertions expressed in the language of 2a. The goal is rules that are simple, without computationally intractable conditions, that enable a large class of interesting assertions to be proved in a relatively straightforward way. We aim to generate proofs automatically via the use of novel type systems (see 4) so these rules will not normally be used directly by users (although see 7a). Understandability is nevertheless important to instill confidence in the integrity of our framework, and the length of proofs is also an important issue in practice (see 9). It will be essential to demonstrate that the rules are sound with respect to the cost model in 1b, and we will also attempt to check relative completeness of certain fragments.

Work carried out: Completed and delivered in September 2003 as Deliverable D2a/b, which presents a VDM-style logic for Grail encoded in Isabelle/HOL together with soundness and completeness proofs and examples that validate the appropriateness of the logic for the purposes of the project. The soundness and relative completeness of the proof rules with respect to the semantics has been proved in Isabelle/HOL; this takes advantage of the decision to develop the proof rules and semantics in Isabelle and provides more confidence in these rules than any informal proof ever could. This work has been written up and submitted for publication [ABH$^+$04].

c. Implement a **proof checker**. *(4 months)*
Deliverables: prototype implementation *Prerequisites: 2b*
Given an ostensible proof built using the rules in 2b, we need to be able to efficiently and automatically check that it is valid. For this purpose, Necula [Nec97] uses Edinburgh LF [HHP93], a generic type-theoretic framework for encoding and checking proofs. Depending on how complicated the logic is, we will take the same approach or alternatively handcraft a proof checker.

Later work (for example 6d) requires that a rudimentary metalanguage be provided, so that lemmas can be proved and then referred to, or for construction of proofs by primitive recursion. Basic facilities of this kind will be provided. The estimate of work required is under the assumption that this will be relatively straightforward. Some care is required to prevent denial of service attacks based on the creation of huge proofs.

Work carried out: Since the proof rules in D2a/b are encoded in Isabelle/HOL, and Isabelle supports exportable proof terms as representations of proofs and contains a proof checking module, we can use Isabelle as a proof checker. Deliverable D2c, delivered in November 2003, explains this and studies the alternatives, taking into account tradeoffs between the size of certificates and the desire to re-use existing software.

d. **Milestone: Completion of implemented logic.** *Prerequisites: 2c*

This has been successfully achieved by the delivery of Deliverables D2a/b and D2c.

e. Implement a **theorem prover**. *(3 months)*
Deliverables: experimental implementation; case studies *Prerequisites: 2b*
Here we plan to experiment with the use of an interactive theorem prover such as Isabelle [Pau94] for generating proofs. This will give access to the features provided by such provers, e.g. context management, goal-directed proof search, and built-in decision procedures. The main point of this work is to explore some of the ramifications of the choice of proof rules in 2b; in practice proofs will be generated by other means, see 4.

Work carried out: Completed and delivered in November 2003 as Deliverable D2e, which describes experiments in the use of Isabelle/HOL for conducting proofs in the bytecode logic. These focus on the use of high-level combinators built on top of the proof rules from D2a/b to support the automatic generation of proofs, as a step towards the automatic generation of certificates, see WP6.

f. Optional: **encode VM semantics** in theorem prover. *(1 month)*
Deliverables: experimental implementation *Prerequisites: 2b*
Here we would encode the entire virtual machine with its semantics in a general-purpose theorem prover such as PVS [ORS92], and then use this to *formally* establish that the proof rules in 2b are sound. The aim would be to explore some of the consequences of our earlier choices in order to validate these decisions. This could be carried out as a final year student project under the supervision of project personnel; this supervision is what is allowed for in the manpower estimate.

Work carried out: Completed and delivered in May 2003 as Deliverable D2f. The virtual machine for the jGrail subset of JVML and its cost model were encoded in Isabelle, along with the translation from Grail to jGrail, and a proof was given that the translation is correct.

The research in this package is a mixture of design, assessment, and implementation work. A working proof checker (2c) will be an essential component part of the final overall system, and the quality of the logic that it implements is vital to the success of the entire framework.

WP3: Design of experimental high-level language

Objectives: Design and implementation of high-level programming language targeted at the bytecode language of WP1, to provide a test bed for WP4–7.

The preceding workpackages provide a grounding for resource guarantees on bytecode; but this is too low a level for practical programming. The objective of this workpackage is to write a compiler for a high-level programming language targeted at the bytecode language of 1a. This will provide a test bed for the higher-level developments of packages 4, 5, 6 and 7. The compiler should be small enough to allow for rapid progress, yet sufficiently expressive to demonstrate that scaling up to real languages is possible.

The most distinctive requirement on the compiler is *transparency*: it must be possible to calculate how a particular fragment of high-level code will transform into bytecode, and what its resulting resource usage will be according to the cost model of 1b. This is vital to support the language-level approach of following workpackages.

As this compiler is to provide a framework for later development, it should also be *extensible*. Other packages aim to add functionality, like resource types (4c) and generation of certificates (6a). This must be possible without disturbing the basic action of the compiler. To meet these requirements the compiler needs an open and well-documented architecture, in sufficient detail to support reasoning (formal and informal) about its behaviour (4e). Consequently, the technical documentation produced by this workpackage is of particular importance, providing reference material for later work.

d. Extend with **immutable objects** and **higher-order functions**. *(4 months)*
Deliverables: prototype implementation; technical documentation *Prerequisites: 3b*
These additional features make the language more expressive, and will assist more advanced parts of later packages, like 7c. Like the base language and compiler, these extensions need comprehensive documentation: covering both the semantics of the language features themselves, and their compilation to bytecode. This particularly important for higher-order functions, as they typically fit less well with existing object-style VM's.

Work carried out: Completed and delivered in November 2003 as Deliverable D3d. The O'Camelot language adds object-oriented programming to Camelot, and in doing so goes considerably beyond the original proposal for immutable objects. It includes objects with state, class and subclass declarations, run-time checked casts, and full interoperation with existing Java libraries, including callbacks. The O'Camelot compiler is available for download from the project website. Several object-oriented examples have been successfully compiled, in O'Camelot and mixed O'Camelot/Java. The deliverable describes a specific example of code for Sun's MIDP platform, to run on handheld devices. It compiles and runs on a PalmOS PDA, making use of the device-specific GUI; it also runs successfully within the Java smartphone emulator testbed. The accompanying technical paper gives semantics, compilation details, and discusses resource implications, and has been accepted for publication [WM04].

e. Implement well-understood **optimisations** in the compiler. *(3 months)*
Deliverables: prototype implementation; technical report *Prerequisites: 3b*

It is sensible to extend the compiler with optimisations whose effect can be determined statically. Example are some uses of in-place update, or the replacement of tail recursion by iteration. Exactly what is possible will depend on the chosen bytecode and its cost model — because of the need to keep track of resource usage in a provable way, this is a tighter constraint than for most compilers.

Work carried out: Completed and delivered in September 2003 as Deliverable D3e. The compiler now performs elimination of tail-recursion and variable consolidation. The technical report includes correctness proofs, and an analysis of the effect on stack usage. For optimisation of heap usage, we have extended the Camelot language with annotations for datatype representation that control the size of objects generated by the compiler. In all cases, we have checked that the "optimisations" do indeed strictly improve space usage of programs.

f. Optional: Incorporate **mutable state** and **concurrency**. *(1 month)*
Deliverables: prototype implementation; technical documentation *Prerequisites: 3d*
Although existing virtual machines provide good support for both of these features, they are less amenable to the kind of formal reasoning presented here. Concurrency in particular has a complex interaction with resource usage. This optional component is closely tied to 7e. This could be carried out as a final year student project, and the manpower estimate is for supervision time.

Work carried out: Completed and delivered in February 2004 as Deliverable D3f, which is a study of the use of O'Camelot for writing stateful multi-threaded programs, while retaining existing resource bounds on individual routines within such code. This introduces new issues concerning resources shared between threads, and the threads themselves as a resource.

WP4: From reasoning principles to high-level type systems

Objectives: Develop reasoning principles and type systems for characterising resource usage, including a typechecker and soundness proofs.

This workpackage builds on the foundational strands in the first three packages. Beginning from the experimental high-level language designed in 3a, we investigate ways of expressing resource constraints and proving that they are satisfied by the compiled program, according to the cost model of 1b. We begin from reasoning principles, perhaps related to the bytecode logic, and then move towards type systems. The notion of type is a very broad one: type-checking can be used to enforce simple consistency checks (that the addition operation + is always applied to two numeric arguments) but also rich semantic notions (such as interference [Rey78], presence of side-effects [TJ92], and resource usage as proposed here or studied in [Hof00]).

Our approach is to stick with type systems where the type-checking problem is *decidable*, whereas the problem of proving that a resource constraint is satisfied will generally be undecidable. This means that we accept an unavoidable gap (the "slack") between the set of programs which are typable in a resource type system and the larger set of programs which satisfy the resource property of interest. For many natural examples, though, the resource bounds are met for obvious reasons which are in the scope of our type systems. The craft of designing type systems lies in capturing these natural examples and minimising the slack, while retaining a practical notion of type and practical type-checking algorithms.

b. Develop a **type system for space-like resources**. *(4 months)*
Deliverables: working design document *Prerequisites: 4a*

The first type system will be one for ensuring space-like resource bounds. There is some relevant recent research here, including [Hof00] which describes a type system with a special *resource type* which corresponds to a unit of reusable space. Hughes and Pareto [HP99] describe a type system for programming in bounded space. Crary and Weirich [CW00] have a type system which provides explicit bounds for time usage: the run time of a function can be expressed as a function of the input. We want to extend and adapt these systems, in particular investigating ways of attaching explicit bounds on space usage. Other resources, such as file handles, network connections, or hardware resources, could be treated in a similar way to space usage in these systems; this will be a novel approach and should help to eliminate a common class of program failures.

Work carried out: Completed and delivered in November 2003 as part of the combined deliverable D4b/c. This describes a space-aware type system for Camelot that is directly inspired by work on LFPL described in [Hof00], with linear use of "diamonds" to account for space usage. Diamond annotations may be added explicitly by the programmer to control re-use manually, or otherwise allocated automatically from a free-list.

In fact, linearity as used in [Hof00] is rather strict and forbids programs which reuse data structures, but nonetheless evaluate correctly within the claimed space bounds. Therefore one direction of our work has been to study and implement more permissive typing systems which analyse aliasing. Konečný and Atkey have worked on typing schemes for analysing sharing, intended as adjuncts to the existing space analysis. Their work can be seen as improvements of the system of Aspinall and Hofmann [AH02], although they each take different approaches.

Konečný's system of *conditions and guarantees* [Kon03] annotates a function with aliasing information based on the structure of datatypes and the data flow within the function. Atkey's type system [Atk04] equips a typing context with information describing relations between resources used by values declared. The main example is regions of memory and the relation of separation.

c. Implement **a typechecker** for the compiler described in 3b. *(4 months)*
Deliverables: prototype implementation *Prerequisites: 3b,4b*
To test our type systems on real examples, we must implement a typechecker for our high-level language. In a production compiler, implementing a typechecker can be a considerable task, especially if there is significant inference involved in type-checking. At the beginning, we will want to separate most of the issues concerning type inference, perhaps requiring extra type annotations in the source language. This means that our type-checking algorithm will be straightforward to implement, especially if we have a *syntax-directed* system where the choice of typing rule is uniquely determined by the syntax of the term we wish to type-check. We will defer more sophisticated type inference until 7d.

Work carried out: Completed and delivered in November 2003 as part of the combined deliverable D4b/c, describing a typechecker for the type system in 4b that has been implemented and integrated with the Camelot compiler described in D3b. A compiler switch turns on/off a simple linearity check which, when engaged, guarantees correct evaluation of programs which use in-place update instructions. The extended Camelot compiler has been successfully run on numerous example programs. Experiments with a Java profiling tool demonstrate a striking correlation between resource typings and actual heap usage: after the initial costs of JVM and GUI initialisation, heap usage follows exactly the pattern of usage predicted by our analysis.

d. **Milestone: Implemented type system for space-like resources.** *Prerequisites: 4c*

This has been successfully achieved by the delivery of Deliverable D4b/c.

e. Prove **soundness over the cost model**. *(6 months)*
Deliverables: technical report *Prerequisites: 4b*
It is natural to ask that our type systems should bear a close relationship to the cost model, so that they can be understood intuitively. Soundness is fundamental: a typing assertion should imply the intended cost constraints, as expressed by our model. Since the cost model applies to the byte code, we must reason about the translations made by the Compiler implemented in 3b. The task here is to conduct a detailed pencil and paper proof to validate this claim.

Work carried out: This soundness requirement can be seen as the combination of "soundness over the bytecode logic" (task 4f) and the soundness of the bytecode logic presented in D2a/b. We are therefore not planning to produce a separate deliverable for 4e.

f. Prove **soundness over the bytecode logic**. *(4 months)*
Deliverables: technical report; conference/journal paper *Prerequisites: 4b*
By the previous part we have a direct connection with the cost model, we know that our type system can also be sound with respect to the bytecode logic for the compiled byte code. In other words, a typing assertion should imply a corresponding proposition in the bytecode logic. The job here is to prove that. It is also something of a test for the bytecode logic, since that is expressed in rather different terms than the type system. Therefore insights from the type system development may lead to tweaks in the proof rules for the bytecode logic, although in general the latter should be stronger (i.e. capable of proving more).

Work carried out: This work is almost complete and a draft of D4f has been produced. Rather than provide an informal argument that the space type system is sound with respect to the bytecode logic, we have worked on formally embedding the type system into the logic, by using a form of "modified assertions" constructed on top of the bytecode logic. The modified assertions are closely related to the type system and its soundness proof (albeit presented at the level of Grail code rather than Camelot).

g. **Milestone: Soundness proofs.** *Prerequisites: 4e,4f*

This has not yet been reached but it will be soon with the completion of D4f.

The research in this package is a combination of improving existing work and combining several different ideas, previously treated in separation and with different motivations. Drawing together these strands is a vital step towards our vision.

WP5: Further high-level and low-level type systems

Objectives: Generalise type systems in WP4 to accommodate more general notions of resource, and develop type systems for expressing resource bounds at the byte-code level.

This workpackage continues and expands on the work begun in 4. The idea here is to begin to generalise the systems for space-like resources studied there to consider more general notions of resource. As a particular case, we will examine type systems for expressing limits on parameter values (5a).

This package also broadens the scope of the type systems to consider *low-level* type systems for expressing resource bounds at the level of the VM byte code. Although our main interest is

in ensuring that resource bounds are met for programming in high-level languages, a way to help ensure this is to push the resource type information as far down as we can, and annotate the bytecode with additional typing information.

a. Develop a type system for expressing **limits on parameter values**. *(4 months)*
Deliverables: technical report; conference paper *Prerequisites: 4a*
As one of the potentially most useful concepts in our generalised concept of resource bounds, we want to have type systems which express restrictions on parameter values for functions. This generalises the most common case needed in programming: that of ensuring that an array access does not violate the bounds of the array. One particularly general way of achieving this is with so-called *dependent types* where the type expression can contain ordinary terms from the language. In practice, full dependent types lead to complex type systems which are almost always undecidable, but restricted forms of dependent types may be useful here. There is interesting related recent research [XH01, Aug98] which may help here. Apart from dependent types, there are other novel type systems for programming languages which propose ideas we might adapt, such as the notion of *shape* [BM97].

Work carried out: Completed and delivered in February 2004 as Deliverable D5a/b, which consists of a book chapter [AH05] together with technical notes. The goal is achieved through the reuse of existing technology including Hongwei Xi's Dependent ML [XP99]. This is illustrated for the case of an embedded automobile control system, where it is critical that a whenever a "brake" procedure be called, its parameters lie within a safety window. It is shown how to represent this with dependent types in a functional language, and so ensure that any program which type checks is guaranteed to satisfy the safety constraints.

b. **Extend soundness proofs** for parameter value constraints *(3 months)*
Deliverables: technical report *Prerequisites: 5a,4e,4f*
The task here is to replay the proofs from 4e and 4f for the new kinds of resource type. This will exercise the generality of our framework; we hope that the earlier proofs will already be amenable to this kind of generalization.

Work carried out: We regard this as completed with the delivery in February 2004 of Deliverable D5a/b. Soundness of this type system is a combination of type soundness of Dependent ML, which is established in [XP99], and the evident soundness of the extension of the bytecode logic described in the deliverable with respect to the extension of the cost model also described there.

c. **Milestone: Type system for parameter value constraints with soundness proofs.**
Prerequisites: 5b

This has been successfully achieved by the delivery of D5a/b.

d. Develop **resource type systems for bytecode**. *(6 months)*
Deliverables: conference/journal paper *Prerequisites: 1a,2b,4a,4b,5a*
Here we want to invent low-level resource type systems for the bytecode itself. There are several reasons to want to do this. First, because the type system will have a closer connection with the machine than the high-level type systems, we can have greater confidence that it truly reflects the resource usage of the machine (especially if later work in 6 and 8 draw on this). Moreover, it establishes a *common resource typing* level, that we might utilise in generalizing the high-level type systems to different languages and language constructs (for example in 7c).

There are also specific formal properties that a bytecode type system enables. A low-level type system makes it possible to state and prove a *type preservation property* for a compiler: that a well-typed high-level language compiles to well-typed bytecode. And we can formalize the *type preservation of bytecode optimisations* using a bytecode type system. In our setting, well-typed means that relevant resource bounds are met, and type preservation will mean that the *same* resource bounds are met.

Work on this subtask will include design of the type systems for the bytecode, perhaps adopting the ideas from 4b and 5a. It could also include formal proofs of soundness for the the bytecode logic, analogous to 4e and 4f.

This subtask is related to existing work on applying high-level programming language type systems to low-level assembly code [MWCG99, CGG+99] and work on type systems for Java bytecode [SA99]. A more direct connection is with recent work on applying space-bound type systems to low-level assembly code [AC03].

Work carried out: We have not begun work on bytecode type systems specifically for Grail and our current plan is to omit tasks 5d and 5e in order to concentrate on work that is more central to the project. However Amadio, who worked with MRG in 2002/2003, has recently demonstrated such a type system for a simplified functional bytecode, building on his work within the MRG project [ACGDZJ04]. This ensures explicit space bounds on execution, based on the quasi-interpretations method of deliverable D4a.

WP6: Generation of certificates

Objectives: Define format of certificates and implement a certificate generator. Experiment with reducing size of certificates.

This package is concerned with generating mobile guarantees of resource boundedness. The guarantee, or *certificate*, is what will be shipped together with the code, as irrefutable evidence for the consumer that the code obeys the desired resource constraints. Here we consider the format of the certificates, and their generation.

a. Implement a **certificate generator**. *(6 months)*
Deliverables: prototype implementation *Prerequisites: 4c*
Given a program which is typed in one of the high-level type systems developed in 4 and 5, we want to automatically generate a certificate which provides manifest evidence of this fact. The certificate contains a proof in our program logic. Here we must design a format for certificates (perhaps based on XML), and implement a software component which generates these from type-checked programs.

Work carried out: Deliverable D6a is close to completion. The precise format of certificates has not yet been defined; this is an essentially straightforward task, but we are postponing it until work on certificate content is finalised. The content of a certificate will consist of an Isabelle representation of the target Grail code, together with typing annotations on methods, which are provided by the compiler based on information output by the type analysis. The compiler will also generate an Isabelle goal representing the claim that the resource typings are valid, together with a tactic invocation that will prove this.

The work in this package is mainly applied research, involving the design and construction of software components. Working software from this package will be an essential part of the final overall system.

Package 9 is dedicated to advanced research topics with the motivation of reducing certificate size. Parts 6c and 6d in this workpackage address this issue. We expect these approaches to be fruitful but not the final word; the results will be useful input for 9.

WP7: Advances in high-level type systems

Objectives: Improve expressiveness, user-friendliness, and accuracy of the type systems developed in WP4 and WP5.

The goal of this workpackage is to improve expressiveness, user-friendliness, accuracy of the type systems developed under 4 and 5.

a. Improving accuracy of type system by **allowing for user interaction**. *(3 months)*
Deliverables: technical report *Prerequisites: 4a,4b,5a*
The aim here is to augment the basic type system from 4 with user annotations in the form of supplied proofs based, for example, on the derived rules from 4a. Also more abstract annotations such as loop invariants could be considered here.

If very restrictive resource bounds are imposed (e.g. hard limits on stack size) we might have to give the programmer the possibility to implement certain critical methods directly in bytecode with corresponding certificates obtained by hand, supported by a theorem prover. The task here will be to enable smooth integration of such user-supplied and -certified routines with high-level code.

Work carried out: The Camelot compiler has been modified to make possible the transfer of user annotations from Camelot programs through to certification generation. However, the language has not yet been extended with user annotations. Although this would be desirable, it is less crucial than planned, in view of our advances with inference mechanisms. It is hoped that this work will be completed as part of an MSc or undergraduate student project.

b. **Adaptation** of bound generation/certification **to optimisations** *(5 months)*
Deliverables: research paper; prototype implementation *Prerequisites: 3e*
Usually, applicability of compiler optimisations (such as those mentioned in 3e) is decided on an ad-hoc basis with correctness of the optimisation being the only criterion. Whether or not an optimisation has taken place is not made visible to the user, it is only through improved overall runtime and space behaviour that the user becomes aware of them.

The purpose of this task is to identify static approximations as to the applicability of certain optimisations and to take their effect into account when calculating resource bounds and certificates. When a tail recursive definition is transformed into an iteration, no memory space for maintaining a stack should be counted when computing resource estimates. Similarly, we must report and appropriately account for the possibility for reusing a temporary file as opposed to creating a new one.

In order to be able to compute resource bounds statically and to retain transparency for the user, it becomes necessary to delineate the applicability of an optimisation by well-defined static criteria. For instance, it will not be sufficient to implement optimisation of tail recursion by discarding the stack frame of the caller at runtime in case the called function is in tail position, because the effect of this on resource usage depends on dynamic aspects known only at runtime. We rather have to syntactically characterise tail recursion and transform the code before actually running it.

Similar but more challenging is the situation with garbage collection. In order to be able to take its effect into account we will have to consider static approximations such as the type system in [Hof00].

This substantially extends the approach to compiler optimisation in [Nec97, PSS98] where it is shown how correctness proofs for an unoptimised program can be transformed into proofs for the optimised program by composing with a general proof that the optimisation is semantics preserving.

In order to demonstrate feasibility it will be sufficient to restrict attention to two representative optimisations, for example tail recursion as iteration and static memory reuse in the style of [Hof00].

Work carried out: No work has been undertaken explicitly on this topic so far, in part because the space-usage inference mechanism takes place at an intermediate stage of compilation, after optimisations have been performed. This means that resource bounds expressed by the inferred types automatically take into account optimisations. Our current plan is therefore to omit this task.

d. **Type inference** *(6 months)*
Deliverables: research paper; implementation (optional) *Prerequisites: 4b,5a,7a,7b*
The basic type system developed under 4 and the extensions developed under 7a and 7b may rely on any number of user-supplied type annotations, for instance, recursive functions might be annotated by suggested bounds on their resource usage which are merely certified, cf. [CW00].

The aim here is to develop ways to infer such annotations automatically to a certain degree based on decision procedures for arithmetic inequalities [HP99], automata-theoretic methods [PWO97], unification [Mil78], program analyses, fixpoint methods [PS91], etc.

Work carried out: Automatic type inference analysis has assumed a more central role than we originally anticipated, and we have made significant research advances on this topic. Type inference following [HJ03] is now a part of the Camelot compiler, and there is also a sophisticated automatic analysis of sharing as described in [Kon03]. Both of these analyses will be used to provide information to the certificate generation phase in WP6.

This workpackage forms part of the scientific core of the proposal. Successful completion will demonstrate that our proposal extends beyond the basic feasibility validated in 1, 2, 4. The goals set out here are ambitious but realistic. In case of difficulty or progress slower than expected it is possible to scale down by e.g. dropping the parts of 7b related to memory management.

WP8: Integration with existing security model

Objectives: Implement resource manager and relate proof-checking infrastructure to present-day security management.

The preceding workpackages have detailed a proof-checking infrastructure which advances the state of the art in security management capabilities. This workpackage will enrich our understanding of this infrastructure by relating it to present-day security management.

b. **Experimental implementation** *(2 months)*
Deliverables: experimental prototype *Prerequisites: 2c,8a,5e,6a*

A prototype implementation of a certificate-led resource manager will be produced. This will provide a platform for further speculative research on developments in static security assessment. Recent work on Java-based agent models which work with the Java 2 security model [GP01] would provide the basis for further development here.

Work carried out: This task, which was scheduled for near the end of year 3, has been brought forward in order to support an early demonstration of what MRG intends to deliver as its main final product, to encourage implementation and integration of project components, and to enhance visibility of the project. At this point it will consist of an integrated web-based demonstration platform that will allow one to receive a program endowed with a certificate of resource usage, check the certificate, and run the program if the check succeeds. The program will be a JVM class file generated by our Camelot compiler and the certificate will be an Isabelle proof script referring to a specialised logic tuned for making statements on space consumption, which builds on our bytecode logic for Grail. Ancillary components in this infrastructure are a driver for the inference of space consumption in Camelot programs, and the automatic generation of Isabelle predicates based on this information. The suitability of our technology for mobile code will be demonstrated at the project review by executing the compiled code on a standard PDA.

WP10: Mobile virtual machines

Objectives: Investigate extension to support downloadable virtual machines.

This workpackage develops a thread of investigation into a mechanism to support mobility between computational environments. As with 9 this investigation is visionary and speculative. Here we are concerned with a promising technology which could provide a way to enable greatly increased interoperability of mobile software. The foundational technology is the *mobile virtual machine*, a bytecode interpreter which is itself downloaded before the bytecode application which is to be interpreted by it. Mobile virtual machines can be realised as *circlets* [Bre98]. One use of this technology would be to allow more advanced virtual machines to be installed between high-level language programs and the JVM. Another would be to perform upgrades on pre-installed micro virtual machines.

a. Understanding the **practical effectiveness** of the technology *(4 months)*
Deliverables: internal technical report *Prerequisites: 1b*

To add another layer of software interpretation to the static virtual machine model calls into question the practical usefulness of this technology in terms of system performance. To consider that the target platform of the mobile virtual machine might be a handheld device further heightens this concern. This task will investigate the use of re-configurable hardware as an implementation technology for this concept. Re-configurable hardware offers the promise of performance close to that of circuitry but without the same specificity.

Work carried out: Completed and delivered in March 2003 as Deliverable D10a, which reached the conclusion that the state-of-the-art in re-configurable hardware and is not sufficiently advanced, and the physical limitations on the technology are sufficiently severe, that it cannot at present be used as a credible technology for virtual machine implementation.

b. A **metalanguage for virtual machines** *(6 months)*
Deliverables: research paper *Prerequisites: 10a,4b,5d*

Another tool to provide partial support for this technology would be a configuration language (or metalanguage) for describing the configuration of next-generation configurable virtual machines. These VMs could then be subject to *just-in-time performance tuning* just before bytecode interpretation by setting or disabling certain optimisation methods. Recent developments in Java technology such as the Java 2 Micro Edition (J2ME) release already define the notion of a *configuration* as a virtual machine and a minimal set of core class libraries and APIs. A J2ME configuration specifies a generalized runtime environment for consumer electronic and embedded devices. Our configuration language would extend this through reference to our virtual machine cost model.

Work carried out: Completed and delivered in February 2004 as Deliverable D10b, which uses the configurable Ant build tool as the vehicle for expressing virtual machine configuration information. This tool is programmed by scripts written in XML, with the definition of the configuration language stated formally as a schema-limited XML file format.

WP11: Project management, dissemination and evaluation

Objectives: Project management, dissemination and evaluation.

The project requires close collaboration between the two sites and provides many opportunities for dissemination. The purpose of this workpackage is to ensure that collaboration proceeds effectively and with attention to internal and external evaluation, while being able to take advantage of a wide variety of forms of dissemination for the results.

The small size of the project enables decisions about the overall technical direction of the project to be taken in close consultation with all of the people involved. Milestones and periodic meetings provide checkpoints where progress can be reviewed and plans adjusted if necessary. Meetings for technical coordination will be as follows:

- A kickoff workshop in month 2 plus workshops in months 11, 23 and 35. These will be attended by all project personnel, insofar as possible. One prominent non-EU expert will be invited to speak at each of the first three workshops at the project's expense; this will give a useful source of comment and advice without the need for project staff to visit these people individually. All of these workshops will be open to people from outside the project, with the final workshop being publicized more widely.

- Internal project meetings in months 7, 17 and 29. Each of these will include technical meetings on all active workpackages, and will be attended by all project personnel involved with those workpackages.

Individual visits are also planned for collaborative technical work.

The results of the project will be made available to all through a website set up at the beginning of the project and maintained under the direction of the Project Coordinator for the duration of the work. This is intended to give interested parties a view of the results as they accumulate. It will include at least the following:

- An introduction to the project including title, partners, and summary, with links to appropriate European Commission websites (GC, FET, IST and/or FP5).

- All of the deliverables and other publications produced by the project, as they are produced. Those that are most appropriate for external consumption will be given special prominence.

- A section (protected from access by non-project personnel) for working drafts, internal project documents, etc.

The results of the project will also be presented at appropriate conferences and published in academic journals. Many of these conferences take place outside the EU and this is taken into account in the travel budget.

The project workshops provide an opportunity for external participants to learn about the project's progress and to contribute their views. The final workshop will be associated with an established international conference for increased visibility; this event is intended more for disseminating the results of the project than for technical coordination and a proceedings is planned. Linking workshops with the annual project evaluation meetings will allow more efficient use of the travel budget.

For self-assessment, each Workpackage Coordinator will supply in advance measurable criteria of progress/success for the different stages of the workpackage which will later be used to assess progress. This assessment of progress will take place in connection with each of the end-of-year project workshops.

a. **Project website** *(1 month)*
Deliverables: website

Work carried out: The project website has been maintained continuously since its establishment. It was thoroughly re-vamped in February 2004.

d. **Workshop at end of year 2** *(0 months)*
Deliverables: workshop

Work carried out: A workshop involving all the projects in Global Computing pro-active initiativeis taking place at Rovereto in March 2004, incorporating reviews of all these projects including MRG; we regard this as completing Deliverable 11d. Although this will give valuable opportunities for cross-fertilization between projects, we do not expect to have sufficient time there for in-depth technical discussion and collaboration between MRG project participants. We are therefore planning an additional internal workshop in Spring 2004.

h. **Assessment of progress in year 2** *(0 months)*
Deliverables: report

Work carried out: Completed and delivered in February 2004 as Deliverable D11h.

Most of the deliverables are allocated 0 person-months because this work will be done by the main investigators rather than the researchers who are employed by the project.

2.3 Comparison of planned activities and actual work

Most of the objectives for year 2 listed in Section 2.1 have been fully achieved. Specifically:

- "Complete the bytecode logic (task 2b) and its implementation (task 2c) and subsequent tasks in WP2 that aim to ensure the quality of the logic and its implementation": **all achieved**

- "Implement a resource type system for heap space bounds for Camelot (tasks 4b and 4c)": **achieved**

- "Implement a function taking a Camelot program with a heap space bound described by a resource typing to a proof in the bytecode logic that the Grail code generated by the Camelot compiler satisfies that bound (see task 6a)": **close to completion**

"Additional priorities were tasks from WP3 (3d and 3e) that were scheduled for year 1 but had not been completed": **achieved**

"... and WP4 tasks from year 2 (4e and 4f)": **close to completion**

"Secondary priorities were tasks in WP5 and WP10 that were scheduled for years 1 and 2": **WP5 most achieved, WP10 achieved**

Some of the tasks that were scheduled for year 2 in the original project workplan are not yet complete, see the table in Section 2.2. Many of these are well underway. Specifically:

- 4e: will be **merged with 4f** since it follows directly from D4f and the soundness proof in D2b
- 4f: is **almost complete**
- 5d: will be **dropped** in order to concentrate on work that is more central to the project
- 6a: is **almost complete**
- 7a: is **low priority** in view of our advances with type inference mechanisms
- 7b: will be **dropped** since our approach to space type inference automatically takes optimisations into account.

Details may be found under each of these tasks in Section 2.2.

In Section 2.6 below we propose some changes to the workplan for year 3, to eliminate some tasks that are peripheral to the core objectives of the project and add tasks that did not appear in the original workplan but now seem to be worthy of effort.

The following charts record the per-workpackage per-site per-annum activity. Manpower is given in person-months and refers only to the researchers paid by the project. The estimated manpower figures are incomplete, since the figures that were required for the Technical Annex were per-workpackage per-site totals, and per-task overall totals, but not the detailed per-workpackage per-site per-annum totals that would be most relevant here. Where it is possible to extract information from the figures in the Technical Annex (for instance, when a Workpackage is scheduled for activity in a single year) then this is recorded in the chart; otherwise the entry is left blank. In such cases, partial information may be obtained by comparing the cumulative totals with the planned totals over the entire project duration.

	Year 1		Year 2		Cumulative		Full project
WP1	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN	9	12	0	0	9	12	9
LMUMUN	4	0	0	0	4	0	4
Total	13	12	0	0	13	12	13

WP1 was finished in year 1. The total amount of effort required was according to plan, but contrary to plan the work was carried out entirely by Edinburgh personnel. Munich effort during year 1 was initially on WP3 and then almost exclusively on WP2 once it became apparent that much more work would be required there than planned.

	Year 1		Year 2		Cumulative		Full project
WP2	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		2		16	7	18	7
LMUMUN		9.5		10	6	19.5	6
Total		11.5		26	13	37.5	13

WP2 is now complete. The effort required was much more than planned, with most of this work (23.5 person-months versus the planned 3 person-months) going into task 2b. This caused major disruption to the planned distribution of work and to the schedule.

	Year 1		Year 2		Cumulative		Full project
WP3	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		3.5		8.8		12.3	9
LMUMUN		5.5		0		5.5	4
Total		9		8.8		17.8	13

WP3 is now (Feb 2004) complete, ahead of schedule. The total amount of effort and the distribution of work between the partners is approximately according to plan.

	Year 1		Year 2		Cumulative		Full project
WP4	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		2		9	8	11	8
LMUMUN		4		2	16	6	16
Total		6		11	24	17	24

WP4 is close to completion. The total amount of effort required has been somewhat more than planned, and the distribution is rather different, with considerable unplanned work in Edinburgh going into space-aware type systems.

	Year 1		Year 2		Cumulative		Full project
WP5	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		0		0		0	9
LMUMUN		1		3		4	9
Total		1		3		4	18

It appears that WP5 will require much less effort than required if the proposal to drop tasks 5d and 5e is accepted, with 5a/b now complete (Feb 2004) and having required less effort than planned. This compensates for the greater amount of work required for some tasks in other workpackages.

	Year 1		Year 2		Cumulative		Full project
WP6	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		0		1.5		1.5	11
LMUMUN		0		9		9	6
Total		0		10.5		10.5	17

WP6 is on track, although more work has been done in Munich than planned and work started later than scheduled.

	Year 1		Year 2		Cumulative		Full project
WP7	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		0		5		5	6
LMUMUN		0		0		0	15
Total		0		5		5	21

Most work on WP7 has been postponed in order to speed up progress on key tasks such as 6a. Some of the progress that has been made here is the work of a PhD student in Munich (Steffen Jost) who is not employed by MRG.

	Year 1		Year 2		Cumulative		Full project
WP8	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN	0	0	0	0	0	0	6
LMUMUN	0	1.5	0	1	0	2.5	2
Total	0	1.5	0	1	0	2.5	8

Work on WP8 has been brought forward from year 3 in order to support an early demonstration of what MRG intends to deliver, to encourage implementation and integration of project components, and to enhance visibilty of the project.

	Year 1		Year 2		Cumulative		Full project
WP9	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN	0	0	0	0	0	0	9
LMUMUN	0	0	0	0	0	0	9
Total	0	0	0	0	0	0	18

Work on WP9 is scheduled for year 3.

	Year 1		Year 2		Cumulative		Full project
WP10	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN		0		3.3		3.3	14
LMUMUN		0		0		0	2
Total	4	0	4	3.3	10	3.3	16

Progress on WP10 is according to schedule. The amount of effort invested has been rather less than planned, with a corresponding decrease in the level of ambition of the tasks, in order to devote effort to tasks that are on the critical path.

	Year 1		Year 2		Cumulative		Full project
WP11	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN	1	0.5	0	0	1	0.5	1
LMUMUN	0	0	0	0	0	0	0
Total	1	0.5	0	0	1	0.5	1

WP11 is proceeding according to schedule. Most of the tasks are allocated 0 person-months because the work is being done by the main investigators rather than the researchers who are employed by the project.

	Year 1		Year 2		Cumulative		Full project
ALL WP	Planned	Actual	Planned	Actual	Planned	Actual	Planned
UEDIN	30	20	30	43.6	60	63.6	90
LMUMUN	24	21.5	24	25	48	46.5	72
Total	54	41.5	54	68.6	108	110.1	163

As discussed in the first progress report, delays in recruiting staff and other circumstances led to a shortage of manpower in year 1, particularly at the Edinburgh site. During year 1 it also became apparent that some key tasks would be much more time-consuming than expected. We responded to this situation by devoting a very large proportion of the available effort to the critical tasks and taking advantage of underspend on personnel in Edinburgh and the availability of well-qualified researchers to recruit additional manpower at that site. The underspend has arisen because researchers were appointed that are more junior than anticipated in the original budget. In particular, a 50% post for a senior researcher at professorial level was included in the budget and was planned to be filled by the Project Coordinator who would be released from 50% of his normal duties to devote this time to technical work on the project, but this turned out to be impossible under Commission rules.

2.4 World-wide 'state-of-the-art' update

This section provides a brief account of particularly relevant technical developments world-wide, which do not already appear in the workplan or previous periodic progress report. We outline each of these developments, giving an evaluation and analysis of their impact on the project.

- At LIF in Marseille, Amadio and others are working on verifying resource bounds for low level code [ACGDZJ04]. Following in part the work done by Amadio within WP4 of MRG [Ama03], they have developed *quasi-interpretations* as a strong technical tool for reasoning about the size of computed values. Rather than the Java target of MRG, they have a custom seven-instruction bytecode, suitable for first-order functional programs. The current state of their project is that given some bytecode with appropriate annotations, they verify that it terminates and runs within specified polynomial space. Separate work considers the problem of automatically synthesising annotations. Although their bytecode is much simpler than Grail, we are interested in their use of termination analysis, and choices for bytecode annotation, to inform the further development of our bytecode logic.

- The SecSafe project is developing static analysis methods for security and safety, with JavaCard as one particular focus [Sec]. This has generated the *Carmel* language, giving a formal operational semantics to model the JavaCard virtual machine [Siv04]. Also working with JavaCard, the firm Trusted Logic (`www.trustedlogic.com`) have released their *Application Diagnosis Tool* which analyses bytecode for conformance to security policies.

- There is continuing work at Sun's research laboratories on the future for Java on very small devices, which naturally has implications for MRG. The Sun *Squawk* system described in [SSB03] is a Java CLDC implementation that runs with a RAM footprint of 1k–8k, and cleanly handles the tripartite memory environment typical of smart cards (RAM, non-volatile memory, and ROM). One of the techniques used is for Squawk to constrain the form of bytecode, in ways very similar to Grail and also to Leroy's work in [Ler02, Ler03]. Such restrictions on control flow and stack use, as investigated in WP1, aid us in precise reasoning about resource usage; for Squawk they simplify verification and garbage collection. In addition, Squawk

limits bytecodes that may trigger garbage collection, and uses region-aware memory management to track pointers between different kinds of memory; both of these should be relevant to improving our own analyses.

- Various groups are working on resource bounds for high-level programming languages. Lee, Yang and Yi [LYY03] have a static analysis approach used to apply a source-level transformation that inserts explicit **free** commands into program text. Some of this resembles Camelot, but it can also generate explicit memory deallocations that are inexpressible in our current linear type system.

  Vasconcelos and Hammond [VH04] present a type system for inferring program run-time, expressed as beta-reductions in a high-level operational semantics. This is more abstract than the explicit clock ticks of the Grail bytecode logic; but their work does apply to full higher-order functions, and may offer us some insights into bounds for higher-order Camelot.

- Work continues on the development of Reynold's separation logic [Rey02]. In particular, recent work of O'Hearn and others [BCO04, OYR04] approaches a general "frame rule" for modular reasoning about pointer structures and hence objects. Calcagno has also demonstrated impressive tools for automating modular reasoning about pointers in certain situations [Cal03]. Inspired by this work, we have experimented under WP4/WP5 with separation logic as an idiom for more advanced proof techniques in the Grail bytecode logic, with encouraging first results.

- Embedded systems face significant resource issues, and relevant to MRG the *Real-Time Specification for Java* [BG+00] describes Java support for programming devices with time and space constraints. It does this by giving the programmer explicit control of scheduling and memory management, rather than formal guarantees; however, we believe that the high-level resource information of Camelot could guide automatic generation of this explicit control code.

- Also with embedded systems, the *Hume* project works on a high-level functional language for safety-critical systems [HM03, MHS04]. This uses strong typing and semantic analysis to generate tight time and space bounds on program execution. Current proposals for the immediate successor to this work involve the Munich MRG group as one partner, and explicitly plan to draw on Grail and the Grail bytecode logic developed in our project [Ham04].

2.5 Clarifications regarding previous review reports

In their report on the first year of MRG the reviewers made several recommendations. We note here the action we have taken in response.

Deliverable D2a was not accepted: The new deliverable D2a/b combines a completely revised D2a with D2b.

Deliverables D2a and D2b were required by 1 Sep 2003: D2a/b was delivered on time. This was a very substantial piece of work and the reviewers were right to be worried about it. In the end, the effort required was 31.5 person-months of paid researchers' time (plus considerable input from the main investigators) instead of the planned 5 person-months. We believe that the result is an extremely firm basis for the many tasks in rest of the project that depend on it. We hope that the reviewers will agree that it addresses all of the concerns in their report.

Submit a reduced and more rational deliverable list: Changes to the workplan are proposed in Section 2.6 below.

MRG should interact more with other GC projects . . . : There has been somewhat increased interaction with other projects, see Section 3 below. However, since we have been behind schedule, work on the tasks in the MRG workplan and publication of results has taken precedence over external interaction. Now that we are almost back on schedule we can devote more effort to this; furthermore we now have much more to offer that is of potential interest to other GC projects than we did a year ago.

. . . in particular PROFUNDIS: The specific suggestion concerning PROFUNDIS was that one of its workpackages has as explicit objective "to develop new type systems to control interference among processes, and to control resource usage". The main thrust of the research of the PROFUNDIS project is concerned with mobility calculi for processes. This can be used to provide guarantees relating to inter-process properties such as non-interference and secure information flow. These can be regarded as resource guarantees, but the resources involved tend to be of a type which are orthogonal to types of resources considered by MRG.

The research which has been performed in the PROFUNDIS project to date is generally much more abstract than that of MRG; PROFUNDIS deals with a very high-level view of mobile computation, whereas MRG is concerned with providing precise bounds on fine-grained resources. Although the stated aims of the two projects are superficially similar there is quite a large gap between the activities of the projects in practise, and it is difficult to find concrete connections between the projects at the present time.

The PROFUNDIS workplan seems to suggest that their work will become more concrete with the progress of time; thus it may be the case that there will a certain amount of convergence with the MRG project, providing better opportunities for collaboration. One example of this is their proposed Task 3.1 of WP3 ("space in types"). They state that they

> ". . . intend to design new, more intensional type operators based on spatial logic operators. As an example of such properties, a type could express the fact that a mobile computation that executes over a host can perform remote outputs only until it does not perform local inputs." [sic]

This looks as if it could potentially be interesting in relation to MRG, in particular to D5a (Type system for expressing limits on parameter values). Unfortunately, it appears that PROFUNDIS have as yet made no progress with this task. However they intend to work on this task in the coming year, and this might be an area where fruitful collaboration would be feasible.

2.6 Planned work for the next reporting period

We propose a number of changes to the workplan for year 3, for several reasons. First, the proposals are intended to address the slippage of the project with respect to the original schedule, which was due to a combination of factors including a manpower shortage in year 1 and some key tasks being much more time-consuming than expected. Second, it is proposed to merge a few closely related tasks in order to give a schedule of deliverables that is less crowded, with a smaller number of larger deliverables. This has already been done with some of the deliverables in year 2 and is proposed above for 4e/f. Finally, we now find that some of the tasks in the original workplan are

peripheral to the core objectives to the project and deserve little or no effort (for example, 5d and 7b above) while others have arisen that deserve to receive attention before the project is complete. The overall aim of these changes are to ensure the production of solid results for the core objectives of the project.

We also propose to extend the project by four months. This will allow participants to disseminate the results of the project at the planned Global Computing event to be held at ETAPS 2005 in April 2005.

The following changes to the workplan are proposed:

- 4e (Proof of soundness over cost model): Merge with 4f into a single task 4e/f entitled "Soundness of type system", since it amounts to a combination of 4f and soundness of the bytecode logic established in 2a/b.

- 5d (Resource type system for bytecode), 5e (Implementation of bytecode type system), 6f (Bytecode typing derivation generator): Omit, as peripheral to the central concerns of the project.

- 6c (Experiment with smaller certificates via formalised soundness proof), 6d (Experiment with smaller certificates via prooflets), 9a (Size-reducing effect of proof-theoretic methods): There is little point to these tasks given the decision to use Isabelle proof scripts in certificates rather than proof terms. Although it is important to the practical use of PCC, the size of proof terms is not a resource-specific issue and we have already done experiments on this as part of D2c. A much more significant gap concerns resource policies and their relationship to the rest of the MRG infrastructure, so add a task to WP8 to study that, see below.

- 7a (Extension of type system by allowing user annotations): Optional task, perhaps as a student project. It is low priority in view of our advances with type inference mechanisms.

- 7b (Adaptation of bound generation/certification to optimisations): Omit, since our approach to space type inference automatically takes optimisations into account.

- 9b (Feasibility of probabilistic certification), 9c (Negotiation-based protocols for resource certification): Merge into a single task 9b/c, entitled "Advanced techniques of certification".

- Add a task to WP8 to address the issue of resource policies:

  8e. Expressing and relating **resource policies**
  Deliverables: technical report *Prerequisites: 6a*
  Our type-system technology will at first allow us to express resource properties of the result of compiling individual functions in the high-level language. We would like to have a way to combine these properties and relate them to user-level statements about the resource usage of a whole program, perhaps dependent on input parameters. An example statement might be "for positive integer inputs n and m, executing the `main()` method requires heap space $32 * n + 16 * m$". This level of description is appropriate for expressing *resource policies* of the code consumer. In this task, we want to investigate ways of expressing and relating resource policies. The aim will be to allow either of two routes towards resource usage checking: generating certificates according to a given resource policy, or attempting to show that a given certificate satisfies an arbitrary resource policy.

- Add tasks to WP5 and WP6 to address the issue of termination, to take account of the fact that the bytecode logic is a logic of partial correctness assertions.

  5g. **Termination checker**
  Deliverables: prototype implementation; technical report *Prerequisites: WP2, WP3*
  The Grail bytecode logic formulates and proves partial correctness assertions. This means that any such assertion will be true for a nonterminating program. Therefore, it is desirable to be able to detect and certify termination as well.

  The reason for decoupling termination from correctness as we have done is that usually different methods are used to prove either so that the two properties are better dealt with in isolation. The alternative would have been a bytecode logic for total correctness with much more complicated rules for method and function invocation.

  To address total correctness we will therefore implement a termination checker for Camelot. We hope to largely be able to use existing technology such as [LJBA03].

  5h. Extension of bytecode logic with **total correctness**
  Deliverables: technical report *Prerequisites: WP3*
  This task is concerned with turning output provided by the termination checker into a formal certificate of total correctness in the bytecode logic. As already mentioned, the bytecode logic in its current state cannot express termination. It must therefore be extended to cover this case. The idea is that we will have both partial correctness assertions and total correctness assertions (program terminates under a precondition and assertion is valid). We will prove a metatheorem allowing one to pass from a partial correctness assertion, obtained e.g. by certificate generation, to the corresponding total correctness assertion in the presence of a proof of termination.

  6g. **Certification of termination**
  Deliverables: prototype implementation; technical documentation *Prerequisites: 5g, 5h, 6a*
  Proofs of termination in the bytecode logic will be automatically generated from the output of the termination checker.

  6h. Optional: **Non-terminating programs**
  Deliverables: technical report *Prerequisites: 6g*
  There are programs that deliberately do not terminate such as user interfaces which take the form of a single infinite loop. In this case one must analyse termination and resource behaviour of the loop body in isolation. In this task we want to investigate this phenomenon in more detail and in particular develop a suggestive example.

2.7 Assessment of project results and achievements

Questions about project's outcomes	No.	Comments or suggestions for further investigation
1. Scientific and technological achievements of the project (and why are they so)		
Question 1.1. Which is the 'Breakthrough' or 'real' innovation achieved in the considered period	N/A	The MRG system as developed so far contains a number of innovations: including the Grail bytecode, its logic, and the Camelot heap management annotations. However, the most distinctive 'Breakthrough' innovation is the work on inferring such annotations using linear programming. Automatic inference for resource types is a leap ahead of expectations, and its implementation within the latest versions of the Camelot compiler show it to be effective and efficient.
2. Impact on Science and Technology: Scientific Publications in scientific magazines		
Question 2.1. Scientific or technical publications on reviewed journals and conferences	32	Details in Section 5
Question 2.2. Scientific or technical publications on non-reviewed journals and conferences	1	Details in Section 5
Question 2.3. Invited papers published in scientific or technical journal or conference.	1	Details in Section 5
3. Impact on Innovation and Micro-economy		
A - Patents		
Question 3.1. Patents filed and pending	0	
Question 3.2. Patents awarded	0	
Question 3.3. Patents sold	0	
B - Start-ups		

Questions about project's outcomes	No.	Comments or suggestions for further investigation
Question 3.4. Creation of start-up	No	
Question 3.5. Creation of new department of research (ie: organisational change)	No	
C - Technology transfer of project's results		
Question 3.6. Collaboration/partnership with a company?	No	
4. Other effects		
A - Participation to Conferences/Symposium/Workshops or other dissemination events		
Question 4.1. Active participation to Conferences in EU Member states, Candidate countries / NAS. (specify if one partner or "collaborative" between partners)	8	Details in Section 5
Question 4.2. Active participation to Conferences outside the above countries (specify if one partner or "collaborative" between partners)	1	Details in Section 5
B - Training effect		
Question 4.3. Number of PhD students hired for project's completion	2	Field: Computer Science
C - Public Visibility		
Question 4.4. Media appearances and general publications (articles, press releases, etc.)	No	

Questions about project's outcomes	No.	Comments or suggestions for further investigation
Question 4.5. Web-pages created or other web-site links related to the project	3	`http://www.lfcs.ed.ac.uk/mrg` MRG project website `http://www.lfcs.ed.ac.uk/camelot` Camelot compiler download site `http://www.lfcs.ed.ac.uk/gc` Global Computing summer school
Question 4.6. Video produced or other dissemination material	0	
Question 4.7. Key pictures of results	0	
D - Spill-over effects		
Question 4.8. Any spill-over to national programs	Yes	Details in Section 3.
Question 4.9. Any spill-over to another part of EU IST Programme	Yes	OpenFET; details in Sections 3 and 5
Question 4.10. Are other team(s) involved in the same type of research as the one in your project ?	Yes	An up-to-date list of related projects is on the project web site.

3 Project management and co-ordination

Cooperation within the consortium has been excellent. The fact that there are only two partners has meant that collaboration is strong.

Project meetings. MRG organized two internal project workshops during year 2, as in year 1. The first workshop was held in Edinburgh during 25–26 May 2003 and involved project personnel and students only. The second workshop was held in Oberschleissheim near Munich during 21–22 November 2003, and involved researchers from Tobias Nipkow's group at the Technical University of Munich and Michal Konečný of Aston University (a former MRG member) in addition to project personnel. Several other "outsiders" were invited to this workshop, including Antoine Galland from the Gemplus Software Research Lab in Marseille, but because of the timing none were able to attend. An open workshop involving all projects in the Global Computing pro-active initiative will be held in Rovereto during 9–12 March 2004 in Rovereto.

The original plan was for one internal workshop and one open workshop per year, with the open workshop being combined with the project review and used partly for dissemination. However, the Global Computing workshops have not provided sufficient opportunity for detailed technical work

and so we have found that an additional meeting to be essential for this purpose.

Collaboration. Most tasks involve contributions from both sites. Most collaboration is via e-mail but intensive periods of joint work have taken place during visits by reseachers at each site to the other site, as follows:

- Beringer to Munich, 27 Jul – 3 Aug 2003
- Loidl to Edinburgh, 10–23 Mar 2003
- Loidl to Edinburgh, 3–17 Sep 2003
- MacKenzie to Munich, 15–23 Feb 2003
- Momigliano to Munich, 21–27 Oct

In addition, in most cases researchers travelling to project workshops spent extra days before or after the workshop at the host site for collaboration.

Personnel changes. The following personnel changes have taken place during year 2:

Roberto Amadio spent a 6-month sabbatical in Munich working on MRG. He returned to Marseille at the beginning of February. He remains in contact and has continued MRG-relevant research, e.g. [ACGDZJ04].

Michel Konečný left the Edinburgh site to take up a lectureship at Aston University at the beginning of September. He continues his involvement with MRG and is helping us to integrate his work on type inference in WP8.

Alberto Momigliano joined the Edinburgh site in July. Since the beginning of November he has been working 50% for MRG and 50% for the University of Milan.

Nicholas Wolverson joined the Edinburgh site in June. Since the beginning of October he has been working 57% for MRG while studying for a PhD.

Brian Campbell joined the Edinburgh site in September, working 57% for MRG while studying for a PhD.

We have been fortunate to recruit such highly qualified researchers. In particular, Momigliano's background in theorem proving using Isabelle and Twelf provides an ideal complement to the expertise of other project members.

Cooperation with other projects.

- Martin Hofmann is coordinator of the APPSEM-II thematic network. APPSEM theme H is "Resource models and web data", which includes MRG, making APPSEM a useful forum for discussion of MRG and related work with other experts.
- There are several points of contact between MRG and the AGILE project. Martin Hofmann and Martin Wirsing (AGILE coordinator, Munich) are conducting preliminary investigations on an extension of the MRG framework to UML using AGILE technology. Don Sannella and Andrzej Tarlecki (AGILE, Warsaw) are collaborating on studying the application of work

from AGILE in the MRG framework, with support from a travel grant funded by the British Council and the Polish Ministry of Scientific Research and Information Technology. David Aspinall is involved in a related investigation with Piotr Hoffman (AGILE, Warsaw).

- Don Sannella is a member of the MIKADO advisory committee.
- Roberto Amadio, who spent 6 months in 2002/2003 working on MRG at the Munich site, is a member of PROFUNDIS. We remain in contact with him since his return to Marseille and he has continued MRG-related research.
- Stephen Gilmore is a member of the DEGAS project. A case study from the DEGAS project (a multi-player on-line role playing game that runs on a Java-enabled mobile telephone) is being used in a student project at Edinburgh which is attempting to build a resource-bounded implementation in Camelot.
- In July, the Edinburgh site organized the EC-sponsored *EEF Global Computing Summer School*. The courses in the summer school were given by Davide Sangiorgi (PROFUNDIS), Martin Wirsing (AGILE coordinator), Rocco De Nicola (AGILE and MIKADO), Martin Hofmann (MRG), Andrew Gordon (Microsoft Research) and Ian Clarke (Freenet). These are all top-ranking researchers and most of them are members of projects in the Global Computing initiative, speaking about research being done within these projects. The summer school was therefore an excellent opportunity to learn about the latest work and to exchange ideas.
- Hans-Wolfgang Loidl and Martin Hofmann are members of a consortium for a STREP proposal under FET Open to study resource bounds in embedded systems.
- MRG continues to have useful interactions with the ConCert project at Carnegie Mellon University as reported last year.
- Members of MRG have been invited to attend CASSIS (Construction and Analysis of Safe, Secure and Interoperable Smart Devices) in Marseille during March, an invitation-only workshop organized by the Everest group at INRIA Sophia Antipolis. This is a prime opportunity for interaction with the main academic and industrial researchers in smart cards, and the invitation amounts to recognition by members of that community that work in MRG is applicable there.
- Don Sannella, Stephen Gilmore and Ian Stark have had discussions with staff at the National e-Science Centre in Edinburgh and with an advisor to the UK e-Science programme on proof-carrying code and resource bounds for grid applications. These discussions have had a (minor) influence on the agenda of e-Science research in the UK.

4 Cost breakdown

See Appendix A.

5 Information dissemination and exploitation of results

Dissemination of the results of MRG has been via publications, research talks and presentations, courses, the project website, software, organization of workshops and joint dissemination activities, and involvement with other projects at national and international level.

Publications. The following publications by MRG members have been appeared or been accepted during 2003.

Invited papers: [HKS03]

Refereed journals and conferences: [ACM03a], [ACM03b], [AL03], [Asp03], [Atk04], [Ber03], [BDG+03], [BDGK03a], [BDGK03b], [BMS03], [CGH+03], [CGH+04], [GHKR03], [GHKR04], [Gil04], [GK03], [HES], [HJ03], [Hof03], [HS], [Jos04], [Kon03], [LDS+03], [MA03], [MHST03], [MP03a], [MP03b], [MT03], [RTL03], [RTL04], [MW04], [WM04]

Non-refereed journals and conferences: [BGHP03]

Research talks and presentations. Talks to present the conference papers listed above, plus the following:

- David Aspinall gave a talk on "Mobile Resource Guarantees" at Warsaw University.
- Stephen Gilmore gave a talk on "Mobile Resource Guarantees for Grid Applications" at the Grids and Applied Language Theory Workshop at the National e-Science Centre, Edinburgh
- Martin Hofmann gave an invited talk on "Certification of memory usage" at the Eighth Italian Conference on Theoretical Computer Science.
- Martin Hofmann gave an invited talk on "Mobile Resource Guarantees" at the Formal Methods for Mobility workshop, Marseille.
- Martin Hofmann gave an invited talk on "A resource-aware program logic" at FGC, Eindhoven.
- Hans-Wolfgang Loidl gave a talk on "A Program Logic for a JVM-like Language" at the Workshop on Hoare Logics, LMU Munich, Munich.
- Hans-Wolfgang Loidl gave a talk on "Mobile Resource Guarantees: Resource Bounds for Functional Languages" at TU Munich.
- Don Sannella gave a talk on "Mobile Resource Guarantees" at the Univerity of Wales in Swansea
- Ian Stark gave a talk on "Mobile Resource Guarantees" in the Analysis of Informatic Phenomena Seminar at Oxford University Computing Laboratory.
- Ian Stark gave a talk on "Mobile Resource Guarantees" at the APPSEM II Workshop in Nottingham.

Courses. Martin Hofmann taught a course on "Type systems for resource control" in the EC-sponsored *EEF Global Computing Summer School* held in Edinburgh during July 2003. This was attended by 50 participants, mostly PhD students from EU countries, see `http://www.lfcs.inf.ed.ac.uk/events/global-computing/`. The lectures covered background material and work being done in MRG, and software produced by MRG was used in laboratory sessions.

Project website. The project website is at `http://groups.inf.ed.ac.uk/mrg/`. It has recently received a thorough facelift.

Software. Grail and Camelot are available for download from the project website, and all other software produced by MRG will also be made available there.

Organization of workshops.

- David Aspinall was organiser and joint Programme Committee chair of the 5th Workshop on User Interfaces for Theorem Provers in Rome.
- Stephen Gilmore was Programme Committee chair for the 4th International Symposium on Trends in Functional Programming in Edinburgh.
- Stephen Gilmore was a member of the Programme Committee of the Implementation of Functional Languages workshop in Edinburgh.
- Martin Hofmann was a member of the Programme Committee of the 7th International Symposium on Functional and Logic Programming in Nara, Japan.
- Martin Hofmann is a member of the Programme Committee of the 6th International Workshop on Implicit Computational Complexity in Turku.
- Martin Hofmann organized a joint TU-Munich/LMU Mini Workshop on Byte Code Logics in Munich.
- Alberto Momigliano was a member of the Programme Committee of the 5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming in Uppsala.
- Don Sannella was a member of the Programme Committee of the Workshop on Foundations of Global Computing in Eindhoven.
- Don Sannella and MRG members in Edinburgh organized the EFF Summer School on Global Computation in Edinburgh.

Involvement with other projects. See Section 3 above.

Exploitation of results. Gemplus, the world's largest maker of smart cards, has shown a strong interest in MRG. Antoine Galland from the Gemplus Software Research Lab in Marseille attended an MRG workshop in September 2002. In his view, MRG offers what seems to be ultimately the most promising approach to the problem of space resource control, which is a critical issue for smart card software because of the severe restrictions on memory size in smart cards imposed by the physical constraints. We have kept up e-mail contact since then and he has been helpful in pointing us to relevant developments and potential examples. We invited him to an MRG workshop in November but he was unable to attend. He has invited us to a meeting at Gemplus in March but there is a direct clash with the MRG review meeting; however Martin Hofmann will meet him and others from Gemplus at the CASSIS workshop in Marseille immediately after the MRG review.

Don Sannella has given short presentations of progress in MRG to researchers from Sun Labs and NTT and to the head of BT Engineering Group. There has been a great deal of interest from NTT in particular, which has a group in California working on applications of proof-carrying code in mobile telephony, but nothing concrete yet. Sun Labs indicates interest in the connection with real-time Java mentioned in Section 2.4 above. Hans-Wolfgang Loidl has discussed MRG with the company AbsInt in Saarbrücken and has collaborated with them on a proposal for a STREP under

FET Open coordinated by the University of St Andrews that will build on MRG results on Grail and the Grail bytecode logic.

References

[ABH+04] David Aspinall, Lennart Beringer, Martin Hofmann, Hans-Wolfgang Loidl, and Alberto Momigliano. A program logic for resource verification. 2004. Submitted for publication.

[AC03] David Aspinall and Adriana Compagnoni. Heap bounded assembly language. *Journal of Automated Reasoning*, 2003. To appear.

[ACGDZJ04] Roberto Amadio, Solange Coupet-Grimal, Silvano Dal Zilio, and Line Jakubiec. A functional scenario for bytecode verification of resource bounds. Rapport LIF 17-2004, Laboratoire Informatique Fondamentale, January 2004. Submitted for publication.

[ACM03a] Simon Ambler, Roy Crole, and Alberto Momigliano. A combinator and presheaf topos model for primitive recursion over higher order abstract syntax. In *Proceedings of the 8th Kurt Godel Colloquium*, Vienna, 2003.

[ACM03b] Simon Ambler, Roy Crole, and Alberto Momigliano. A definitional approach to primitive recursion over higher order abstract syntax. In *Proc. MERLIN 2003*, 2003.

[AH02] David Aspinall and Martin Hofmann. Another type system for in-place update. In *Proc. 11th European Symposium on Programming, Grenoble*, volume 2305 of *Lecture Notes in Computer Science*. Springer, 2002.

[AH05] David Aspinall and Martin Hofmann. Dependent types. In Benjamin Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 4. MIT Press, to appear in 2005.

[AL03] David Aspinall and Christoph Lüth. Proof General meets IsaWin. In *Proc. User Interfaces for Theorem Provers 2003 (UITP'03)*, September 2003. Available from `http://www.informatik.uni-bremen.de/uitp03/`.

[Ama03] Roberto Amadio. Max-plus quasi-interpretations. In *Proceedings of the 6th International Conference on Typed Lambda Calculus and Applications*, number 2701 in Lecture Notes in Computer Science, pages 31–45. Springer-Verlag, 2003.

[Asp03] David Aspinall. Type checking parametrised programs and specifications in ASL+FPC. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Recent Trends in Algebraic Development Techniques, 16th International Workshop, WADT 2002, Frauenchiemsee, Germany, 2002, Revised Selected Papers*, LNCS 2755. Springer, 2003.

[Atk04] Robert Atkey. A calculus for resource relationships. In *SPACE 2004: Second workshop on Semantics, Program Analysis, and Computing Environments for Memory Management*, Venice, 2004.

[Aug98] Lennart Augustsson. Cayenne — a language with dependent types. In *ICFP '98: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming*, pages 239–250. ACM Press, 1998.

[BCO04] Richard Bornat, Cristiano Calcagno, and Peter O'Hearn. Local reasoning, separation and aliasing. In *SPACE 2004: Second workshop on Semantics, Program Analysis, and Computing Environments for Memory Management*, Venice, 2004.

[BDG+03] L. Brodo, P. Degano, S. Gilmore, J. Hillston, and C. Priami. Performance evaluation for global computation. In C. Priami, editor, *Global Computing: Programming environments, languages, security, and analysis of systems. Proceedings of the IST/FET International Workshop (GC 2003)*, volume 2874 of *LNCS*, pages 229–253, Rovereto, Italy, February 2003. Springer-Verlag.

[BDGK03a] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Derivation of passage-time densities in PEPA models using IPC: The Imperial PEPA Compiler. In G Kotsis, editor, *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pages 344–351, University of Central Florida, October 2003. IEEE Computer Society Press.

[BDGK03b] J.T. Bradley, N.J. Dingle, S.T. Gilmore, and W.J. Knottenbelt. Extracting passage times from PEPA models with the HYDRA tool: A case study. In S. Jarvis, editor, *Proceedings of the Nineteenth annual UK Performance Engineering Workshop*, pages 79–90, Warwick, July 2003.

[Ber03] Lennart Beringer. A programming language based analysis of operand forwarding. In *Proceedings of the 12th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'03)*, number 2860 in LNCS, L'Aquila, 2003. Springer-Verlag.

[BG+00] Greg Bollella, James Gosling, et al. *The Real-Time Specification for Java*. Addison-Wesley, 2000.

[BGHP03] L. Brodo, S. Gilmore, J. Hillston, and C. Priami. Mapping coloured stochastic Petri nets to stochastic process algebras. In P. Kemper, editor, *On-site proceedings of the ICALP Workshop on Stochastic Petri Nets and Related Formalisms*, pages 47–66, Eindhoven, Holland, 2003. University of Dortmund Research Report number 780.

[BM97] G. Bellè and E. Moggi. Type intermediate languages for shape-analysis. In *Typed Lambda Calculi and Applications: Proceedings of the Third International Conference TLCA '97*, number 1210 in Lecture Notes in Computer Science, pages 11–29. Springer-Verlag, April 1997.

[BMS03] Lennart Beringer, Kenneth MacKenzie, and Ian Stark. Grail: a functional form for imperative mobile code. In *Foundations of Global Computing: Proceedings of the 2nd EATCS Workshop*, number 85.1 in Electronic Notes in Theoretical Computer Science. Elsevier, June 2003.

[Bre98] Gordon Brebner. Circlets: Circuits as applets. In *IEEE Symposium on FPGAs for Custom Computing Machines*, April 1998. Napa Valley, California.

[Cal03] Cristiano Calcagno. A low complexity fragment of separation logic. First APPSEM-II Workshop, Nottingham, UK, March 2003.

[CGG+99] K. Crary, N. Glew, D. Grossman, R. Samuels, F. Smith, D. Walker, S. Weirich, and S. Zdancewic. TALx86: A realistic typed assembly language. In *1999 ACM SIGPLAN Workshop on Compiler Support for System Software Atlanta, GA, USA*, pages 25–35, May 1999.

[CGH+03] C. Canevet, S. Gilmore, J. Hillston, M. Prowse, and P. Stevens. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, 150(2):107–120, March 2003.

[CGH+04] C. Canevet, S. Gilmore, J. Hillston, L. Kloul, and P. Stevens. Analysing UML 2.0 activity diagrams in the software performance engineering process. In *Proceedings of the Fourth International Workshop on Software and Performance*, pages 74–78, Redwood Shores, California, USA, January 2004. ACM Press.

[CW00] K. Crary and S. Weirich. Resource bound certification. In *Proc. 27th Symp. Principles of Prog. Lang. (POPL)*, pages 184–198. ACM, 2000.

[GHKR03] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. PEPA nets: A structured performance modelling formalism. *Performance Evaluation*, 54:79–104, 2003.

[GHKR04] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo. Software performance modelling using PEPA nets. In *Proceedings of the Fourth International Workshop on Software and Performance*, pages 13–24, Redwood Shores, California, USA, January 2004. ACM Press.

[Gil04] S. Gilmore. Extending camelot with mutable state and concurrency. In *Proceedings of the International Conference on Computational Science (ICCS'04)*, LNCS, Kraków, Poland, June 2004. Springer-Verlag.

[GK03] S. Gilmore and L. Kloul. A unified tool for performance modelling and predicition. In *Proceedings of the 22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP'03)*, number 2788 in LNCS, pages 179–192, Edinburgh, Scotland, September 2003. Springer-Verlag.

[GP01] Stephen Gilmore and Marco Palomino. BabylonLite: Improvements to a Java-based distributed object system. In *Proc. 4th CaberNet Plenary Workshop, Pisa*, 2001.

[Ham04] Kevin Hammond, coordinator. EmBounded: Automatic prediction of resource bounds for embedded systems. FET Open STREP project proposal, January 2004.

[HES] Martin Hofmann, Martin Escardo, and Thomas Streicher. On the non-sequential nature of the interval-domain model of exact real-number computation. *Mathematical Structures in Computer Science*.

[HHP93] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *JACM*, 40(1):143–184, 1993.

[HJ03] Martin Hofmann and Steffen Jost. Static prediction of heap space usage for first-order functional programs. In *Proceedings of the 30th ACM Symposium on Principles of Programming Languages*, New Orleans, 2003.

[HKS03] Jo Hannay, Shin-ya Katsumata, and Donald Sannella. Semantic and syntactic approaches to simulation relations. In *Proc. 28th Intl. Symp. on Mathematical Foundations of Computer Science*, volume 2747 of *Lecture Notes in Computer Science*, pages 68–91. Springer, 2003.

[HM03] Kevin Hammond and Greg Michaelson. Hume: a domain-specific language for real-time embedded systems. In *Generative Programming and Component Engineering: Proceedings of GPCE 2003*, number 2830 in Lecture Notes in Computer Science, pages 37–56, September 2003.

[Hof00] Martin Hofmann. A type system for bounded space and functional in-place update. *Nordic Journal of Computing*, 7(4):258–289, 2000.

[Hof03] Martin Hofmann. Linear types and non size-increasing polynomial time computation. *Information and Computation*, 183:57–85, 2003.

[HP99] J. Hughes and L. Pareto. Recursion and dynamic data structures in bounded space: towards embedded ML programming. In *Proc. International Conference on Functional Programming (ACM). Paris, September '99.*, pages 70–81, 1999.

[HS] Martin Hofmann and Philip Scott. Realizability models for BLL-like languages. *Theoretical Computer Science*.

[Jos04] Steffen Jost. `lfd_infer`: an implementation of a static inference on heap space usage. In *Proceedings of Second Workshop on Semantics, Program Analysis and Computing Environments for Memory Management (SPACE 2004)*, 2004.

[Kon03] Michal Konečný. Functional in-place update with layered datatype sharing. In *Proceedings of the 6th International Conference on Typed Lambda Calculus and Applications*, number 2701 in Lecture Notes in Computer Science, pages 195–210, Valencia, 2003. Springer-Verlag.

[LDS+03] H-W. Loidl, F. Rubio Diez, N.R. Scaife, K. Hammond, U. Klusik, R. Loogen, G.J. Michaelson, S. Horiguchi, R. Pena Mari, S.M. Priebe, A.J. Rebon Portillo, and P.W. Trinder. Comparing parallel functional languages: Programming and performance. *Higher-Order and Symbolic Computation*, 16(3):203–251, 2003.

[Ler02] Xavier Leroy. Bytecode verification for Java smart card. *Software Practice & Experience*, 32:319–340, 2002.

[Ler03] Xavier Leroy. Java bytecode verification: Algorithms and formalizations. *Journal of Automated Reasoning*, 30(3–4):235–269, 2003.

[LJBA03] Chin Soon Lee, Neil D. Jones, and Amir Ben-Amram. The size-change principle for program termination. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, London, 2003.

[LYY03] Oukseh Lee, Hongseok Yang, and Kwangkeun Yi. Inserting safe memory reuse commands into ML-like programs. In *Proceedings of the 10th Annual International Static Analysis Symposium*, number 2694 in Lecture Notes in Computer Science, pages 171–188. Springer-Verlag, 2003.

[MA03] Alberto Momigliano and Simon Ambler. Multi-level meta-reasoning with higher order abstract syntax. In *Proc. 6th Intl. Conf. on Foundations of Software Science and Computation Structures*, Warsaw, 2003.

[MHS04] Greg Michaelson, Kevin Hammond, and Jocelyn Serot. FSM-Hume: Programming resource-limited systems using bounded automata. In *SAC 2004: Proceedings of the 19th Annual ACM Symposium on Applied Computing*. ACM Press, 2004. To appear.

[MHST03] Till Mossakowski, Anne Haxthausen, Donald Sannella, and Andrzej Tarlecki. CASL—the common algebraic specification language: Semantics and proof theory. *Computing and Informatics*, 22:285–321, 2003.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, August 1978.

[MP03a] Alberto Momigliano and Frank Pfenning. Higher-order pattern complement and the strict lambda-calculus. *Transactions on Computational Logic*, 4(4):493–529, 2003.

[MP03b] Alberto Momigliano and Jeff Polakow. A formalization of an ordered logical framework in hybrid with applications to continuation machines. In *Proc. MERLIN 2003*, 2003.

[MT03] Alberto Momigliano and Alwen Tiu. Induction and co-induction in sequent calculus. In *Post-Proceedings of TYPES workshop*, 2003.

[MW04] Kenneth MacKenzie and Nicholas Wolverson. Camelot and Grail: resource-aware functional programming for the JVM. In *Trends in Functional Programming*, 2004. To appear.

[MWCG99] G. Morrisett, D. Walker, K. Crary, and N. Glew. From System F to typed assembly language. *ACM Transactions on Programming Languages and Systems,*, 21(3):528–569, May 1999.

[Nec97] George Necula. Proof-carrying code. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1997.

[ORS92] S. Owre, J. Rushby, and N. Shankar. PVS: a prototype verification system. In *Proc. 11th Intl. Conf. on Automated Deduction, Springer LNCS vol. 607*, pages 748–752, 1992.

[OYR04] Peter W. O'Hearn, Hongseok Yang, and John C. Reynolds. Separation and information hiding. In *POPL 2004: Conference Record of the 31st Annual ACM Symposium on Principles of Programming Languages*. ACM Press, 2004.

[Pau94] Lawrence C. Paulson. *Isabelle — A Generic Theorem Prover*. Lecture Notes in Computer Science 828. Springer-Verlag, 1994.

[PS91] Jens Palsberg and Michael Schwartzbach. Object-oriented type inference. In *Proc. ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 246–161, 1991.

[PSS98] Amir Pnueli, Michael Siegel, and Eli Singerman. Translation validation. In *Proc. of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98), LNCS 1384*, pages 151–166, 1998.

[PWO97] Jens Palsberg, Mitchell Wand, and Patrick O'Keefe. Type inference with non-structural subtyping. *Formal Aspects of Computing*, 9:49–67, 1997.

[Rey78] J. C. Reynolds. Syntactic control of interference. In *Proc. Fifth ACM Symp. on Princ. of Prog. Lang. (POPL)*, 1978.

[Rey02] John Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS 2002: Proceedings of the Seventeenth Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, 2002.

[RTL03] Andre Rauber Du Bois, Phil Trinder, and Hans-Wolfgang Loidl. Mobile computation in haskell. In *Proc. 12th Intl. Workshop on Functional and (Constraint) Logic Programming*, Valencia, 2003.

[RTL04] A. Rauber Du Bois, P.W. Trinder, and H-W. Loidl. Implementing mobile Haskell. In *Trends in Functional Programming*, 2004. To appear.

[SA99] Raymie Stata and Martin Abadi. A type system for Java bytecode subroutines. In *ACM Transactions on Programming Languages and Systems 21*, volume 21(1), January 1999.

[Sec] SecSafe. Secure and safe systems based on program analysis. IST Project 1999-29075. `http://www.doc.ic.ac.uk/~siveroni/secsafe/`.

[Siv04] Igor Siveroni. Formalisation of the operational semantics of the Java Card virtual machine. *Journal of Logic and Algebraic Programming*, 58(1-2):3–25, January 2004.

[SSB03] Nik Shaylor, Douglas N. Simon, and William R. Bush. A Java virtual machine architecture for very small devices. In *Language, Compiler, and Tool Support for Embedded Systems: Proceedings of LCTES '03*, number 38(7) in ACM SIGPLAN Notices, pages 31–41, July 2003.

[TJ92] Jean-Pierre Talpin and Pierre Jouvelot. The type and effect discipline. In *Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California*, pages 162–173, Los Alamitos, California, 1992. IEEE Computer Society Press.

[VH04] Pedro Vasconcelos and Kevin Hammond. Inferring costs for recursive, polymorphic and higher-order functional programs. In *IFL 2003: Proceedings of the 15th International Workshop on the Implementation of Functional Languages*, Lecture Notes in Computer Science. Springer-Verlag, 2004. To appear.

[WM04] Nicholas Wolverson and Kenneth MacKenzie. O'Camelot: Adding objects to a resource aware functional language. In *Trends in Functional Programming*, 2004. To appear.

[XH01] Hongwei Xi and Robert Harper. A dependently typed assembly language. In *Proc. 6th ACM SIGPLAN Intl. Conf. on Functional Programming*, pages 169–180, Florence, September 2001.

[XP99] Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 214–227, San Antonio, January 1999.

A Financial summary

	UEDIN		LMUMUN		TOTAL	
	Year 2	Total	Year 2	Total	Year 2	Total
Personnel, estimated	174128	341561	114156	223934	288284	565495
Personnel, actual	131319	196518	119146	216649	250465	413167
Durable equipment, estimated	0	12495	0	7990	0	20485
Durable equipment, actual	4248	5782	2096	10090	6343	15872
Subcontracting, estimated	0	0	0	0	0	0
Subcontracting, actual	0	0	0	0	0	0
Travel and subsistence, estimated	17080	32160	9720	22440	26800	54580
Travel and subsistence, actual	13807	23011	11150	13338	24957	36349
Consumables, estimated	1130	2260	1110	2219	2240	4479
Consumables, actual	1868	3070	0	0	1868	3070
Computing, estimated	6107	12214	4500	9000	10607	21214
Computing, actual	4950	4950	4491	8982	9441	13932
Protection of knowledge, estimated	0	0	0	0	0	0
Protection of knowledge, actual	0	0	0	0	0	0
Other specific costs, estimated	2400	15380	4260	5460	6660	20840
Other specific costs, actual	0	559	0	0	0	559
Subtotal, estimated	200845	416070	133746	271043	334591	687113
Subtotal, actual	156193	233890	136883	249059	293076	482949
Overheads, estimated	40169	83214	26749	54208	66918	137422
Overheads, actual	31239	46779	27377	49812	58615	96591
Total, estimated	241014	499284	160495	325251	401509	824535
Total, actual	187431	280669	168146	298870	371107	579539