DELIVERABLE SUMMARY SHEET

Project Number:	IST-2001-33100
Project Acronym:	PROFUNDIS
Title:	Proofs of Functionality for Mobile Distributed Systems

Deliverable Number:	8
Due date:	Project month 24
Delivery date:	February 1, 2004

Short Description: Static methods for program transformation in presence of stack inspection; for static control of migrating processes; for specifying the state transitions of a protocol (session types); a generalized and integrated type system for resource bounds; formalization in a proof assistant of a method to guarantee noninterference; type systems for Klaim and for a large fragment of the Java Virtual Machine. Integration of type systems to the verification tools developed in workpackage 1.

Partners owning:	INRIA
Partners contributed:	FFCT, INRIA, Pisa, UU
Made available to:	

Contents

Nota: The title of this deliverable, ***Integration and algorithms***, conforms to what was announced in the TA, but does not faithfully reflect the results obtained in Year 2. Indeed, almost all our results go into the *integration* aspect, having mostly worked on the extension, generalisation and enhancent of the novel type systems we had introduced in Year 1. The work on algorithms has not brought results yet, but we plan to address this issue in Year 3.

1 Summary

Globally, the work in WP 3 has proceeded according to plans. As expected, all four tasks in the TA have been addressed, thus testifying that Year 2 is a year of transition between the definition of novel type systems (Tasks 3.1 and 3.2), that was at the heart of Year 1's work, and the study of the applicability of these type systems (Tasks 3.3 and 3.4).

In deliverable 8, we present the results obtained this year. In the case of one subtask (type systems for secure composition of components), we have adopted an approach that departs from what was expected, exploiting a graph-based static analysis instead of a traditional type system.

We also discuss the discrepancies in some domains w.r.t. what was announced. The subtasks that did not lead to significant results in Year 2 are: *space in types*, *type-inference algorithms*, and, to a partial extent, *advanced programming constructs* and *approximation techniques.* This is justified either by unexpected technical difficulties (in the last case), or by organisational questions that caused some delays.

We do not regard these difficulties as alarming because:

- the quality and quantity of our results in the other subtasks are satisfying, sometimes going beyond what was expected;
- we are confident that the difficulties mentioned above can be tackled next year.

2 Detailed description of work

A substantial part of the work carried in Year 1 has been developed along the path planned in the TA and SUP, moving from Tasks 3.1 and 3.2 to Tasks 3.3 and 3.4, i.e., from the definition of new type systems towards the study of their generalizability and their applicability.

The work on *'resource allocation'* (Task 3.1) that had been started in Year 1 in the context of the calculus of Mobile Ambients has been generalized and developed, thus moving to Task 3.2 (*'Integration of type systems'*). [9] presents a generalisation of the original type system for Mobile Ambients that handles flexible resource control policies, departing from the original approach based on counting the number of processes physically present at a given site. In [10], the robustness of this method was tested by porting the mechanisms for resource control from the calculus of Mobile Ambients to related formalisms for distributed computation such as Boxed Ambients, Nomadic Pict, and the Kell calculus (these models are developed in other projects belonging to the GC initiative).

Another work related to the integration of type systems is presented in [5], where session types are used in the context of the λ-calculus with side-effecting input/output to specify and verify complex protocols. This work is related to [11], where session types are used to reason on objects and components.

The question of access control (Task 3.1) has been studied from the point of view of static control of migrating processes in a distributed framework. We have defined a type system for $lsd\pi$, an explicitely distributed version of the π-calculus. The role of this type system is to control remote communication, process migration and channel creation, in order to guarantee compliance w.r.t. a control policy on migration of code. This work is reported in [7].

The study of type systems for non-interference properties (Task 3.1) has been developed, and some of the issues mentioned in Tasks 3.3 and 3.4 have been tackled. The entry *'Advanced programming constructs'* of Task 3.3 has been addressed by defining a type system for a large fragment of the Java Virtual Machine [3]. The formalization in the Isabelle/HOL proof assistant of a type system for a concurrent language with scheduling, based on work by Boudol and Castellani, has been described in [2]. This contribution, that goes in Task 3.4 (*'case studies and tool developments'*), represents, to the best of our knowledge, the first machine-checked account of non-interference for a concurrent language.

The work on using types to guarantee properties of termination of processes (Task 3.1) has continued. The main people involved have been Davide Sangiorgi and Yuxin Deng. Last year we had experimented with techniques based on logical relations, trying to transplant proof techniques well-established in funtional languages onto the pi-calculus. During this year we have extended that work to richer processes calculi, including features like higher-order communication and distribution. This is however merely an extension – we have not been able to add new substantial ideas. We have also experimented a different direction, where termination is guaranteed by means of a combination of types with techniques from term rewriting. We are currently writing a paper on the results obtained.

A. Ravara et al. have worked on typing concurrent objects in process calculi, by introducing an Algebra of Behavioural Types [8]. This work goes in Task 3.2 (Integration of types with operational and logic techniques). In this approach, a type is given by a higher-order labelled transition system. This transition system is then used, via the introduction of a notion of bisimulation, to give an internal view of objects inhabiting a given type. Several algebraic laws, as well as completeness results for image-finite types, are obtained.

Davide Sangiorgi and Yuxin Deng have been working on enhancements of algebraic and operational techniques for untyped mobile processes using type information (Task 3.2). A paper on the results obtained is in preparation.

In contrast with the aforementioned lines of research, the entry *'Secure composition of components'* of Task 3.1 has been addressed in an original way, that, technically, does not follow what was originally planned in the TA. In the context of access control policies based on stack inspection and dynamic security policies, as can be found in security-aware languages such as Java or C♯, a method based on *control flow graphs* has been introduced to validate some program transformations without compromising security [4]. The use of control flow graphs has turned out to be an appropriate alternative to more traditional type systems for the programs being analysed.

3 Sum up per task

We recall below the tasks and subtasks, as given in the TA, and present our contributions in Year 2.

- Task 3.1: Interference and access control
  - Classification of interferences: finished in Year 1
  - Resource allocation: types for migration control [7]
  - Declassification: this subtask is obsolete (see Year 1's Deliverable number 3)
  - Secure composition of components: work on session types [5, 11]; stack inspection and program transformations [4]
- Task 3.2: Integration of types with operational and logic techniques
  - Operational and logical techniques for typed calculi: behavioural types [8]; also work in progress by Johansson (UU, on the addition of types to the verification tools) and Y. Deng and D. Sangiorgi (on behavioural properties in the context of types for termination)
  - Expressiveness: no result yet
  - Space in types: no result yet.

- Integration of type systems: generalization and porting to different calculi/languages of previous work on types for resource allocation [9, 10]
- Approximation techniques: work in progress by Y. Deng and D. Sangiorgi on types for termination of mobile processes

- Task 3.3: practicality and scalability

  - Type-inference algorithms: no result yet.
  - Advanced programming constructs: security types for the Java Virtual Machine [3, 1]; the KLAIM project [6]

- Task 3.4: Case studies and tool development

  - Case studies: no result yet.
  - Implementations: formalization of a type system to guarantee non-interference [2]

4 Plans for Year 3

In Year 3, we plan to continue the ongoing work on *operational and logical techniques for typed calculi* and on *approximation techniques*, and to publish results on (at least some of) these topics. These contributions could in particular be helpful for the sake of verification algorithms (Workpackage 1).

Subtasks that have already led to published results but that we plan to investigate further include *secure composition of components* (where the work on stack inspection can be adapted to standard compilers) and *integration of type systems* (where the approach for resource allocation control shall be ported to a 'standard' version of the π-calculus).

Finally, we want to put an important effort in addressing those subtasks that have not led to results yet, namely *expressiveness* and *space in types* (Task 3.2), *type-inference algorithms* (Task 3.3) and *case studies* (Task 3.3).

5 Comments

The papers [2, 9, 7] present some work that had been done mostly during Year 1. Other papers present new work, developed in Year 2.

References

[1] G. Barthe, A. Basu, and T. Rezk. Security Types Preserving Compilation. In *Proc. of VMCAI'04*, LNCS. Springer Verlag, 2004. to appear.

[2] G. Barthe and L. Prensa Nieto. Formally Verifying Information Flow Type Systems for Concurrent and Thread Systems. manuscript.

[3] G. Barthe and T. Rezk. Secure Information Flow for the JVM. manuscript.

[4] M. Bartoletti, P. Degano, and G. Ferrari. Stack Inspection and Secure Program Transformations. *International Journal of Information Security*, 2003. To appear.

[5] Simon Gay, Vasco T. Vasconcelos, and Antnio Ravara. Session types for inter-process communication. TR 2003–133, Department of Computing, University of Glasgow, March 2003.

[6] L.Bettini, V.Bono, R.De Nicola, G.Ferrari, D.Gorla, M.Loreti, E.Moggi, R.Pugliese, E.Tuosto, and B.Venneri. The klaim project: Theory and practice. In C.Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, number 2874 in LNCS, 2003.

[7] F. Martins and A. Ravara. Typing migration control in lsdpi. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2003. Submitted for publication.

[8] A. Ravara, P. Resende, and V. Vasconcelos. An algebra of behavioural types. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2003. Submitted for publication.

[9] D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources (long version). *International Journal of Information Security*, 2003. to appear.

[10] David Teller. Formalisms for mobile resource control. *Electronic Notes in Theoretical Computer Science*, 85(1), 2003. Presented at FGC'03.

[11] A. Vallecillo, V. T. Vasconcelos, and A. Ravara. Typing the behavior of objects and components using session types. *Electronic Notes in Theoretical Computer Science*, 68(3), 2003. Presented at FOCLASA'02.

WP3 Scientific Contributions

APPENDIX

Appendix 1.1.1
F. Martins and A. Ravara. Typing migration control in lsdpi. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2003. Submitted for publication.

Appendix 1.1.2
Simon Gay, Vasco T. Vasconcelos, and Antnio Ravara. Session types for inter-process communication. TR 2003–133, Department of Computing, University of Glasgow, March 2003.

Appendix 1.1.3
A. Vallecillo, V. T. Vasconcelos, and A. Ravara. Typing the behavior of objects and components using session types. *Electronic Notes in Theoretical Computer Science*, 68(3), 2003. Presented at FOCLASA'02.

Appendix 1.1.4
M. Bartoletti, P. Degano, and G. Ferrari. Stack Inspection and Secure Program Transformations. *International Journal of Information Security*, 2003. To appear.

Appendix 1.2.1
A. Ravara, P. Resende, and V. Vasconcelos. An algebra of behavioural types. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2003. Submitted for publication.

Appendix 1.2.2
D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources (long version). *International Journal of Information Security*, 2003. to appear.

Appendix 1.2.3
David Teller. Formalisms for mobile resource control. *Electronic Notes in Theoretical Computer Science*, 85(1), 2003. Presented at FGC'03.

Appendix 1.3.1
G. Barthe and T. Rezk. Secure Information Flow for the JVM. manuscript.

Appendix 1.3.2
G. Barthe, A. Basu, and T. Rezk. Security Types Preserving Compilation. In *Proc. of VMCAI'04*, LNCS. Springer Verlag, 2004. to appear.

Appendix 1.3.3
L.Bettini, V.Bono, R.De Nicola, G.Ferrari, D.Gorla, M.Loreti, E.Moggi, R.Pugliese, E.Tuosto, and B.Venneri. The klaim project: Theory and practice. In C.Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, number 2874 in LNCS, 2003.

Appendix 1.4.1
G. Barthe and L. Prensa Nieto. Formally Verifying Information Flow Type Systems for Concurrent and Thread Systems. manuscript.