PROJECT PERIODIC REPORT

DRAFT VERSION

Grant Agreement number: 243381
Project acronym: CERCO
Project title: Certified Complexity
Funding Scheme: STREP
Date of latest version of Annex I against which the assessment will be made:

Periodic report: 1st X 2nd ☐ 3rd ☐ 4th ☐
Period covered: from 01 February 2010 to 31 January 2011

Project coordinator: Prof. Claudio Sacerdoti Coen
Alma Mater Studiorum – Università di Bologna
Tel: +39 051 2094973
Fax: +39 051 20 9 4510
E-mail: claudio.sacerdoticoen@unibo.it
Project website address: http://cerco.cs.unibo.it

Table of contents

Declaration by the scientific representative of the project coordinator

I, as scientific representative of the coordinator of this project and in line with the obligations as stated in Article II.2.3 of the Grant Agreement declare that:

The attached periodic report represents an accurate description of the work carried out in this project for this reporting period;
The project (tick as appropriate) [1]:

X has fully achieved its objectives and technical goals for the period;

has achieved most of its objectives and technical goals for the period with relatively minor deviations.

has failed to achieve critical objectives and/or is not at all on schedule.

The public website, if applicable

X is up to date

is not up to date

To my best knowledge, the financial statements which are being submitted as part of this report are in line with the actual work carried out and are consistent with the report on the resources used for the project (section 3.4) and if applicable with the certificate on financial statement.
All beneficiaries, in particular non-profit public bodies, secondary and higher education establishments, research organisations and SMEs, have declared to have verified their legal status. Any changes have been reported under section 3.2.3 (Project Management) in accordance with Article II.3.f of the Grant Agreement.

Name of scientific representative of the Coordinator: Dr. Claudio Sacerdoti Coen

Date: 24/02/2011

For most of the projects, the signature of this declaration could be done directly via the IT reporting tool through an adapted IT mechanism.

[1] If either of these boxes below is ticked, the report should reflect these and any remedial actions taken.

3.1 Publishable summary

Project context and objectives

The CerCo project (Certified Complexity) aims to construct a formally verified complexity preserving compiler from a large subset of C to some typical microcontroller machine language, of the kind traditionally used in embedded systems.

The work consists of the definition of cost models for the input and target languages, and the machine-checked proof of preservation of complexity (concrete, not asymptotic) along the compilation chain. In particular, the compiler will also return tight and certified cost annotations for the source program (expressed in terms of clock-cycles) for program slices with O(1) complexity. It is then up to the user, possibly assisted by automatic tools, to use this (trusted) information to state and to prove precise complexity assertions on the program. To this aim, he can exploit the tools provided by ongoing research related to techniques for invariant generation and cost inference for imperative programs, with two additional benefits: a guarantee that the inferred intensional properties will carry over to assembly code; and the adoption of a cost model that is absolutely precise, being induced from the generated assembly language.

The main focus of the current project is on the certified, cost annotating compiler. As for the way cost annotations are used, we shall develop a proof-of-concept prototype, by interfacing with existing tools, to show how this information can be exploited in a useful manner. We will apply abstract interpretation techniques to automatically infer complexity bounds and, in particular, we will test these techniques on the C code generated by the compilers of synchronous languages, such as Lustre or Esterel.

The major breakthrough of the project is the possibility of giving a tight performance estimate for executable code (for computing platforms such as microcontrollers, not relying on operating systems), a task that is currently regarded as highly visionary in the compiler community. The essential paradigm shift consists in creating the (certified) infrastructure allowing to draw conclusions on the target code, while comfortably reasoning on the source.

The project also complements work done by the Worst Case Execution Time (WCET) community that focuses on sophisticated techniques for computing approximate costs of assembly (actually, object code) programs. In WCET the focus is usually on more complex microprocessors than we do, and the analyses often mix together static and empirical analyses. The most sophisticated attempts try to relate estimated costs for the object code to the intermediate or source languages of programs, as we do. This is usually done with external tools that are not integrated with the compiler and that are not certified. On the contrary, we do the cost inference as part of the compilation, and will be fully certified. A combination of the two methodologies to address most complex architectures is surely possible, but it is not envisioned as part of the project.

The compiler will be open source, and all proofs will be public domain. More information on CerCo can be found at the project Web site: http://cerco.cs.unibo.it.

Work performed so far and main results achieved

During the first period of the project we focused on the problem of conceiving a methodology for building a compiler which can lift in a provably correct way pieces of information on the execution cost of the object code to cost annotations on the source code. To this end, we needed a clear and flexible picture of: (i) the meaning of cost annotations, (ii) the method to prove them sound and precise, and (iii) the way such proofs can be composed. We have examined two approaches to these three questions which we have named direct and labelling.

In the direct approach, a cost is assigned to source, intermediate and target instructions, and we show preservation of costs (up to approximations) at each compilation step. In this approach, introducing approximations seems unavoidable. Moreover, the proof associated to each compilation step is very algebraic in nature, having to show that upper bounds for the costs are met.

In the labelling approach, the operational semantics of the source, intermediate and target languages are augmented to record traces of execution. The information stored in the traces roughly records the number of times each control block is executed. The proof shows that the traces are essentially preserved during compilation. The proof is quite symbolical, since no algebraic computation is performed on the traces. No approximation is introduced to preserve the traces. Thus, any cost that is computed on the target code and that is associated only to a trace is automatically lifted to the source language without loosing precision.

The two methodologies have been applied to a toy compiler for evaluation, and the proof of preservation of complexity has been partially verified with a proof assistant. This formal study has suggested that the labelling approach, unlike the direct one, has good compositionality and scalability properties.

In parallel with the development of the labelling approach, we have assembled a moderately optimizing compiler from a large subset of C to the Mips assembly language. The compiler is written in O'Caml and is based on a compilation chain close to the one considered in the CompCert project, which represents the state of the art in compiler certification, but whose proofs are not free software. The compiler has been used in the context of a master-level compilation course at UPD.

We have implemented and tested the labelling approach on top of this prototype compiler and we have produced experimental evidence that the labelling approach can handle a realistic C compiler with a moderate level of optimization (comparable to level 1 optimization of GCC).

Following these preliminary and encouraging experiments, the decision has been taken to target a processor with a more industrial flavour and which is widely used in embedded software. The choice has fallen on the MSC-51 processor. The MCS-51 is an 8-bit microprocessor introduced by Intel in the late 1970s. Commonly called the 8051, in the decades since its introduction the processor has become a popular component of embedded systems. The processor, its successor the 8052, and derivatives are still manufactured en masse by a host of vendors. The 8051 has a relatively straightforward architecture, unencumbered by advanced features of modern processors, making it an ideal target for formalisation.

The port of our O'Caml prototype compiler from Mips to the MCS-51 is now completed. This represented the first milestone for CerCo. From now on, the development of CerCo will follow two parallel routes.

The first one consists of formalizing the prototype using the interactive theorem prover Matita and then certifying the prototype itself. We will formally prove in Matita both the preservation of the extensional semantics and the correctness of propagation of cost information.

The second route will consist in integrating the compiler with existing tools in order to exploit cost information. The general picture is as follows: once the program is compiled, the user will obtain precise cost information for the execution of O(1) program fragments. Then, manually or by means of invariant generators, he will provide precise or approximated cost invariants, depending on the application. Finally, proof obligations will be generated form the cost invariants by means of the weakest precondition method. We also plan to provide ad hoc tactics to solve the simple and common proof cases associated with cost obligations.

The decisive fact that has led to the selection of this processor is the availability of precise timing information. The MCS-51 is also interesting because of its memory model that is not at all uniform. The MCS-51 has several distinct memory regions characterized by a different access time and different sizes. In particular, some important memory regions, as the one used for the internal stack, are so small that the programmer has to take care so as to not exhaust memory. This is usually achieved by granting the C programmer some control overs memory allocation. For instance, the SDCC free software compiler provides type annotations for variables and pointers to specify the memory region where variables should be allocated to, or where pointers should point to.

The semantics of these high level extensions are not trivial, especially from the point of view of costs associated to casts. For instance, the values of a generic pointer pointing to an unspecified memory region are made of an actual pointer whose size is region dependent and a tag to identify the region. Dereferencing a generic pointer requires some case analysis over the tag and casting some dynamic checks. In CerCo we have given a precise semantics to these extensions by specifying in Matita an executable operational semantics for the extended version of the C-Light language. We also completed the semantics with labels to compute cost traces. This is a preliminary step to the certification of our compiler. A formal semantics for C-Light was already described in Coq in the CompCert project, but there the semantics was not executable. To gain confidence in our semantics, we first ported the CompCert formalisation to Matita and we then formally proved the equivalence of the two semantics when no extension is given. Executability allows grants us further confidence by running programs inside Matita (or extracting the code to O'Caml) and comparing the program behaviour with the expected one.

The final, preliminary, step of the formalisation of our compiler has been giving a formal semantics to the MCS-51, which was not present in the literature. In order to compute a precise evaluation cost, the formalisation is done at the machine code level, and not at the assembly level. All the fetch-decode-execute cycle is described. Actually, we provided two distinct versions, both executable, one in O'Caml and the other in Matita.

The O'Caml formalisation captures the whole MCS-51 ecosystem: not only the logical operations, but also I/O, the UART, a variety of timers and interrupts, an assembly language extended with labels and pseudo-instruction, an assembler that also maps generic jumps to jumps to offsets of a particular size (short, medium and large jumps) and finally a parser and pretty printer for the Intel HEX format used to store and load code images. Labels are also considered, so that we can compute cost traces.

The Matita formalisation is currently less well developed. At the moment, the formalisation in Matita only captures those features that are required to certify our compiler. In particular, it does not capture I/O, interrupts, the UART and the parser and pretty printer for Intel HEX.

The next steps towards the formalisation of the compiler will be: the formalisation of executable semantics for all the intermediate languages; the total rewriting of the compiler code in Matita, exploiting dependent types to capture as many invariants as possible; the proof of correctness itself.

Expected final results

In spite of the many recent, astonishing achievements in proof checking and program verification, the field of computer assisted reasoning, with its long term vision of a fully dependable, formally checked information society is still one of the most visionary fields of computer science. The impressive results obtained in this area are only partially due to the advancements in tools for assisted reasoning, but also to a growing confidence in their potential.

Compilers, filling the gap between humans and machines from the programming point of view, are one of the main tools at the base of information technologies. We cannot hope to build truly trustworthy, dependable and long-lasting systems over shaky foundations: the semantics of compilation must be better understood. There is some history of work in certified compilers, and a series of workshop meetings on the topic. Only very recently has end-to-end certification of a realistic compiler been attempted in the CompCert project, and several projects on this topic are starting to target more complex scenarios.

This previous work is focused on denotational aspects of computation. Our proposal is new in addressing intensional aspects, such as preserving space or time bounds of the source in compiled code. This is still perceived by the compiler community as a highly visionary goal, that would provide a major milestone in this area.

Our proposal contributes to the long-term community goal of formally checked reliable computer systems both directly, by our original work on a certified complexity preserving compiler, and more generally by large scale use of state-of-the-art tools and techniques for software certification. Our proposal takes such technologies past the proof-of-concept phase to a maturity level that could facilitate a substantial degree of technology transfer to industry.

The direct, long term, impact of the project on real-time systems (in particular reactive systems) will be that of dramatically increasing the trust on response time of the systems, while at the same time simplifying code by removing some of the checks for approaching deadlines. Moreover, we envision a future where compilers (certified or not) will give back to the user guarantees and information on the intensional behaviour of the produced binaries (like time, space and power consumption), to be exploited by semi-automated reasoning tools.

The certified compiler developed in the project is just a first step in this direction. Being the first example of a compiler that provides intensional guarantees, we do not expect to be able to immediately exploit cost annotations in an effective way. This requires an effective convergence of the compiler implementation, proof assistance and invariant generation communities. Moreover, further work is required to target processors for non-embedded systems (which implement many hardware optimizations) and to enlarge the class of inter-procedural and loop optimizations that the compiler can handle whilst tracking intensional properties. Together with the need for better understanding the issues related to multiple backends for different architectures, the time-frame required to bring this technology into the mainstream can be estimated at around a decade.

It would certainly be interesting to address advanced features of general purpose processors such as cache memory. We believe that such a task will be naturally achieved by integrating certified compilers with the tools provided by the WCET community that, at the moment, are very powerful, but lack any sort of formal guarantee.

To summarize, the project responds to the increasing expectation for trustworthy, dependable and long-lasting systems by developing reliable compilers not only from the point of view of behaviour but also of performance. Building certified compilers (combined with a machine checked proof of their semantics preservation) is an emerging topic whose relevance is destined

to grow in coming years. It appears to be of high importance for the ICT industry to have methods and tools for producing certifiable systems, and automatic checking of complex invariants (such as preservation of complexity) is a critical step and a major scientific challenge.

3.2 Project objectives for the period

3.2.1 Overview

The project aims to construct a formally verified complexity preserving compiler from a large subset of C to some typical microcontroller machine language, of the kind traditionally used in embedded systems. The work is comprised of the definition of cost models for the input and target languages, and the machine-checked proof of preservation of complexity (concrete, not asymptotic) along the compilation chain. In particular, the compiler will also return tight and certified cost annotations for the source program (expressed in terms of clock-cycles) for program slices with O(1) complexity. It is then up to the user, possibly assisted by automatic tools, to use this (trusted) information to state and prove precise complexity assertions about the program.

The major novelty of the project is the possibility of giving tight performance estimates for the executable code (for computing platforms such as microcontrollers, not relying on operating systems), a task that is currently regarded as highly visionary in the compiler community. The essential paradigm shift consists in creating the (certified) infrastructure allowing to draw conclusions on the target code, while comfortably reasoning on the source code.

During the first period of the project (first year only) we had three major objectives:

- the development of an untrusted cost-annotating O'Caml compiler (**Deliverable D2.2**):
  a proof of concept untrusted prototype of the cost annotating compiler, written in the O'Caml programming language. Preliminary to this was the choice of the global architecture for the compiler, together with the choice of intermediate languages (**Deliverable D2.1**).
- the choice of source and target languages for the compiler (also part of D2.1), together with an executable formalisation (in the interactive proof assistant Matita) of their operational semantics (**Deliverables D3.1 and D4.1**).
- the development of a Web site for CerCo (**Delievrable D6.1**), together with the infrastructure required to manage the project and foster collaboration and reuse of results between the project members. The plan for use and dissemination of foreground (**Deliverable D6.2**).

The first objective, in the form of Deliverable D2.2, also represents our only milestone for the first period. The prototype is essential for the continuation of the project, since:

- It serves as a skeleton of the actual compiler to be implemented in Matita (in the second period) and formally verified (in the second and third period). The differences between the Matita and O'Caml compiler is caused by the difference in the two programming languages and by the need to rearrange code to simplify the formalisation.
- It is the starting point of the study of the management of complexity obligations (second period). Complexity obligations will be generated starting from complexity invariants that are user provided invariants based on the cost annotations generated by our compiler. The prototype, being already able to generate cost annotations, allows one to attack the problem of management of annotations, invariants and proof obligations on realistic examples.

The core of the work in the second objective consists of anticipating in the first period as much as possible the work on formal verification. Moreover, an executable formal semantics for a widespread language (the C subset we consider) and for a realistic microprocessor (our target) are especially valuable by themselves and potentially useful to third parties.

3.3 Work progress and achievements during the period

3.3.1 Progress overview and contribution to the research field

As clearly visible in the Pert diagram in Annex 1, the workplan for CerCo has been designed to maximize parallelism between two important activities.

The first activity, performed in WP2 and WP5, consists of the development of an untrusted framework for the analysis of intensional properties of programs written in C. The important landmarks for this activity are:

1) the development of the untrusted cost annotating O'Caml compiler;
2) the integration of the compiler in a larger framework that allows one to manage the machine-provided cost annotations and the human-provided cost invariants;
3) the integration within the framework of procedures to automate the trivial proof cases commonly found in proofs of complexity obligations;
4) the study of some use cases

Landmark 1) was successfully attained during the first period of the project. Landmark 2) is the target for the next period.

Landmark 1), which is also the project milestone M1, already is a significant achievement in the domain of compiler design. Indeed, this constitutes the first example of a compiler that is able to induce absolutely precise cost annotations on the source language. As far as we know, in the compiler construction literature there is no comparable work for comparative assessment.

The Worst Case Execution Time (WCET) community has developed sophisticated techniques for computing approximate costs of assembly (actually, object code) programs. They usually focus on more complex microprocessors than we do, and their analyses often mix together static and empirical analyses. The most sophisticated attempts try to relate estimated costs on the object code to the intermediate languages or to the source languages of programs, as we do. They usually do this with external tools that are not integrated with the compiler and that are not certified. On the contrary, we do the cost inference as part of the compilation, and will be fully certified.

With our approach we finally obtain a fully certified compiler that preserves at once the extensional and the intensional semantics of the program. However, we pay the price of not being able to exploit all optimisations of existing compilers, and our compiler design will be less modular.

The second activity, performed in WP3 and WP4, consists of the formalisation of the core component of the framework, the cost annotating compiler itself. The important landmarks for this activity are:

1) the formalisation of the source language and the target architecture
2) the formalisation of the intermediate languages used during compilation

3) the rewriting of the untrusted compiler in Matita
4) the certification of the rewritten compiler

Landmark 1) has been successfully attained during the first period of the project. Landmarks 2) and 3) are the targets for the next period.

In the literature there already exists several formalisations of microprocessors and assembly languages. Usually, the formalisers are focused on the certification of the microcode or the gate level of processors and are their formalisations are therefore largely different from the formalisations used in compiler verification. The latter, instead, stop at the assembly level, and this does not allow one to capture intensional aspects of the execution of the processor.

What we provided is a formalisation of both the microprocessor and the assembly level, combined with an assembler. This allows us to lift the intensional properties of machine language programs to the assembly level, keeping track of the assembling strategy. Moreover, we do not only formalize the ALU of the micro-processor, but the whole ecosystem of the MCS-51 architecture, comprising I/O and timers, capturing precisely the intensional behaviour that we are interested in.

Considering the formalisation of the source language, which is a subset of C, we can also see another advance over the state of the art. Instead of considering a completely idealized and uniform memory model, we modelled an abstraction of a memory model that incorporates the existence of distinct memory regions with different properties (access time, pointers size, etc.). Moreover, we gave a formal semantics for ad hoc extensions of the C language commonly used in programming microprocessors with non-uniform memory models (e.g. the ad hoc extensions of the SDCC compiler for MCS-51 microprocessors). While these extensions are often used, as far as we know they have never been given a formal semantics in the literature, though they are necessary to apply formal methods to programs that exploits these extensions.

3.3.2 Work packages progress

WP2: Untrusted compiler prototype

The goal of this Work Package is to implement a proof-of-concept prototype for the cost annotating compiler. The compiler will be untrusted, meaning that no proof will be given that the machine code and the cost annotations returned by the compiler are correct. It will be written in a high-level, comfortable programming language particularly tailored to compiler construction (O'Caml). This untrusted prototype compiler will drive the design and implementation of the trusted version, and at the same time will allow us to start experimenting with the management of cost annotations, the declaration of complexity assertions, the generation of complexity obligations and their interactive solution (tasks covered by WP5).

During the first period, three tasks have been active: Task 2.1, Task 2.2 and Task 2.3.

Task 2.1 was devoted to architectural design of the CerCo compiler. The main issue to be solved in this task was to devise not only an architecture for a cost-preserving compiler, but also a proof strategy for the correctness of cost inference and for the preservation of costs.

The most significant result was the identification of such a proof strategy, which was tested on a “toy” compiler. The proposed strategy emerged from some more straightforward, but failing

strategies tried in advance. The proposed strategy, and the reasons for failure of the other ones, are detailed in Deliverable D2.1.

Task 2.2 was devoted to the choice of the intermediate languages for the compiler. Here we opted for a conservative choice, mimicking the ones previously used in the CompCert project. Concretely: 1) we reused CompCert intermediate languages for the front-end; 2) we took from a CompCert inspired Pascal compiler by Francois Pottier the intermediate languages for the backend; 3) we augmented the length of the compilation chain by adding another intermediate, target independent language at the interface between the front-end and the back-end. This is meant to delay instruction selection, which is performed quite early in CompCert. The intermediate languages are also detailed in D2.1 ``Compiler design and intermediate languages''.

Task 2.3 was devoted to the implementation of the Untrusted Cost-Annotating O'Caml Compiler (Deliverable D2.2), which represents the most significant result of the task and also of the first period of the project. The implementation was actually performed in two stages: first we developed a MIPS cost annotating prototype compiler with unrealistic cost inference performed on the assembly code. Then we ported the prototype to our target architecture, the MCS-51 microprocessor family. This was done in parallel with the formalisation in O'Caml of the MCS-51 (Task 4.1) and also with the formalisation of the MCS-51 related superset of the C subset considered in the project (Task 3.1). Priority has been put in obtaining a functional compiler without slipping out of schedule. For this reason, at the moment the prototype only addresses the C subset and not the MCS-51 specific extensions designed in Task 3.1 which were not considered in the project proposal (Annex 1). Similarly, at the moment the inference of costs performed by the prototype is unrealistic. The reason is that, to obtain realistic costs, the costs must be inferred from the machine code and not from the assembly (with pseudo instructions) that is the compiler's target. An assembler for our assembly language has been provided in Task 4.1, now allowing us to compute the realistic costs. The code that implements the cost inference has already been written, but at the moment it is undergoing testing. For this reason we decided to not integrate it yet in the final prototype (Deliverable D2.2).

Task 2.4, not active in the first period of the project, is for integration, validation and testing of the compiler developed in Deliverable D2.2. In particular, D2.2 will be modified to reflect changes to the code suggested during the formalisation. We plan to exploit this task to also integrate in the compiler the realistic cost inference and possibly also the MCS-51 specific extensions. However, integration of the latter, which was not originally planned in Annex 1, will be given lower priority to avoid delays in the schedule.

As already, explained **the only deviation from Annex 1 for WP2 is the lack of integration of the realistic cost inference engine** and this can be easily fixed in Task 2.4 by simply integrating the code already written, but untested at the moment.

WP3: Verified Compiler – Front End

The goal of this Work Package is to build the trusted version of the compiler front-end, from some abstract syntax tree representation of (a large subset of) the C language to three-address like intermediate code. During the first period, only Task 3.1 has been active. The aim of task 3.1 is

to provide a formal, executable semantics of the C subset in Matita. This is an important preliminary step for the certification of the CerCo compiler, to be started in the second period.

The most significant result is Deliverable D3.1, the executable formal semantics of the C subset. As a starting point, we chose to use C-Light as our subset of C. C-Light has already been adopted by several tools working on C has a de-facto simplified representation of C programs. The simplification is mainly related to the parsing phase, where some normalization of the code is performed. In place of using the standard C-Light, we have extended it with ad hoc modifications for variable and pointer declarations. These non-standard modifiers are commonly implemented by compilers for microprocessors used in embedded systems. They allow the programmer to instruct the compiler in the choice of memory region used for variables (e.g. internal, small but fast memory versus external larger but slower memory). Since internal memory and external memory have different sizes in the MCS-51 architecture, the one targeted by our compiler, the modifiers also determine the size of pointers to data stored in memory regions. In order to allow users to exploit these modifiers in their own programs, we have modified the C to C-Light parser to propagate the modifiers.

Before the start of our formalisation, there existed another free software formalisation of the C-Light semantics in Coq, developed as part of the CompCert project. That formalisation constituted the core of our formalisation in Matita. However, we can spot two main differences. The first one is related to the memory model. While the model of Coq assumes a uniform memory model made of disjoint segments all addressed by means of integers, our memory model is much more complicated, since we have to support several kind of regions addressed with pointers of different sizes. The second difference is about executability. While the semantics in Coq is given by means of inference rules, we provide a directly executable semantics inside Matita. In order to augment our confidence in the correctness of our semantics, we have also proved the equivalence between our executable semantics and the one given with inference rules.

We do not deviate from Annex 1 for WP3.

WP4: Verified Compiler – Back End

The goal of this Work Package is to build the trusted version of the compiler back-end, from intermediate three address code to assembly language.

During the first period, only Task 4.1 has been active. The aim of task 4.1 is to provide a formal, executable semantics of machine code in Matita. This is an important preliminary step for the certification of the CerCo compiler, to be started in the second period.

The most significant result is Deliverable D4.1, the executable formal semantics of the MCS-51 micro-processor architecture. We have actually provided two different formalisations. The most complete one is in O'Caml. The O'Caml formalisation comprises all aspects of the MCS-51 architecture: the ALU, timers, I/O, the UART and interrupts. We also provided an assembly language with labelled memory areas, jumps to labels and pseudoinstructions for jumps of unknown size. An assembler function has been implemented, together with parsing and pretty-printing functions from/to the Intel HEX format, to allow interoperability with third party tools.

The formalisation also captures the highly unorthogonal memory model of the MCS-51, made of several distinct kinds of memory addressed by means of several addressing mode, that are in many-to-many relation with the kinds of memory. We also provided cost labels for our assembly language to trace the cost of control blocks of our assembly program, and relate the actual cost of the object code with the one at the assembly level.

The second formalisation has been carried out in Matita and it will be used to formally prove the correctness of the compiler. Compared to the O'Caml formalisation, it does not yet capture timers, I/O and interrupts. At the moment, we do not plan to address these features in our compiler, handling all I/O via low level routines directly coded in assembly. Hence we do not need to formalize these features in Matita to prove the correctness of the compiler.

The completion of the Matita formalisation w.r.t. The O'Caml one is left as low priority future work. The main difficulties related to the Matita formalisation with respect to. the O'Caml formalisation are due to the lack of programming constructs (mainly phantom types and polymorphic variants) used in O'Caml to cleanly capture the unorthogonality of the instruction set. Dependent types have been used as often as possible in the Matita formalisation and they have provided alternative solutions for the same problems, practically avoiding the need for polymorphic variants and phantom types.

We do not deviate from Annex 1 for WP4.

WP5: Interfaces and Interactive components

The aim of WP5 is to develop a proof of concept prototype, by interfacing to already existing tools, to show how the annotations produced by the compiler can be exploited to draw complexity assertions on the execution time of the program.
No task of WP5 was active during the first period.

WP6: Dissemination and exploitation

The overall objective of WP6 is to manage the knowledge generated by the project and IPRs, and to bring the technological advances developed within the CerCo project to the scientific community and potential users. The project will target not only the scientific and academic communities but also European industries potentially interested in applying formal verification techniques to embedded software design.
The specific objectives of WP6 will be:

(1) a tailored dissemination activity that will make use of specific dissemination mechanisms in order to reach the relevant communities;
(2) supervision of the entire project with regard to result applicability and the promotion of the exploitation.

Task 6.1, user validation and exploitability, was not active in the first period.
Task 6.2 is about the contribution to portfolio and concertation activities at FET-Open level.

The dissemination activity performed in the first period is described in Section 5.2.

3.4 Deliverables and milestones tables

Deliverables

Table 1. Deliverables

Del. no.	Deliverable name	Version	WP no.	Lead beneficiary	Nature	Dissemination level[2]	Delivery date from Annex I (proj month)	Actual / Forecast delivery date	Status	Contractual	Comments
D6.1	Project Web Site and Software Repository	1.0	6	UNIBO	P	PU	3	15/06/2010	Submitted	Yes	
D2.1	Compiler Design and Intermediate Languages	1.0	2	UPD	R	PU	6	10/09/2010	Submitted	Yes	
D6.2	Plan for the use and dissemination of foreground	1.0	6	UNIBO	R	CO	6	10/09/2010	Submitted	Yes	

[2] **PU** = Public
PP = Restricted to other programme participants (including the Commission Services).
RE = Restricted to a group specified by the consortium (including the Commission Services).
CO = Confidential, only for members of the consortium (including the Commission Services).
Make sure that you are using the correct following label when your project has classified deliverables.
EU restricted = Classified with the mention of the classification level restricted "EU Restricted"
EU confidential = Classified with the mention of the classification level confidential " EU Confidential "
EU secret = Classified with the mention of the classification level secret "EU Secret "

D3.1	Executable Formal Semantics of C	1.0	3	UEDIN	P	PU	10	16/12/2010	Submitted	Yes	
D4.1	Executable Formal Semantics of Machine Code	1.0	4	UNIBO	P	PU	10	16/12/2010	Submitted	Yes	
D2.2	Untrusted Cost-Annotating Ocaml compiler	1.0	2	UPD	P	PU	12	16/02/2011	Submitted	Yes	
D1.1	Periodic Activity Report and Financial Statements	1.0	1	UNIBO	R	CO	12	31/03/2011	Non submitted	Yes	This document,.

PU = Public
PP = Restricted to other programme participants (including the Commission Services).
RE = Restricted to a group specified by the consortium (including the Commission Services).
CO = Confidential, only for members of the consortium (including the Commission Services).
Make sure that you are using the correct following label when your project has classified deliverables.
EU restricted = Classified with the mention of the classification level restricted "EU Restricted"
EU confidential = Classified with the mention of the classification level confidential " EU Confidential "
EU secret = Classified with the mention of the classification level secret "EU Secret "

Milestones

Table 2. Milestones							
Milestone no.	Milestone name	Work package no	Lead beneficiary	Delivery date from Annex	Achieved	Actual / Forecast achievement date	Comments
MS1	Untrusted Cost-annotating compiler	2	UPD	31/01/2011	Yes	16/02/2011	

3.5 Project management

3.5.1 Management activities

CerCo is a relatively small project with three partner and a small number of work packages. Management activities are included in WP1, whereas WP2, 3, 4 and 5 correspond to research activities.

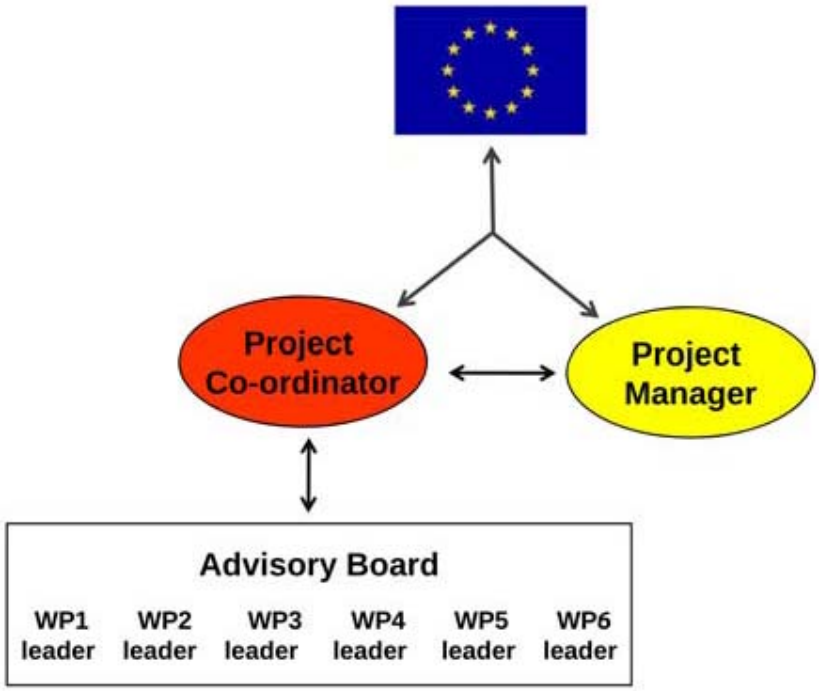
The overall objective of WP1, as indicated in Annex I of GA is the smooth implementation of the CerCo project and in particular:

- to coordinate and supervise activities to be carried out;
- to carry out the overall administrative and financial management of the project;
- to manage contacts with the European Commission;
- to monitor quality and timing of project deliverables;
- to establish effective internal and external communication procedures.

Consortium management tasks and achievements

Consortium structure

The project management and administrative structure is described in Annex I of GA Section B2.1. No changes occurred with respect to the management structure. For convenience, a graph illustrating the project management structure is reproduced below.



Financial Management and Coordination

The coordinator UNIBO was in charge of the overall administrative and financial tasks for CerCo. The activities during this first period included:

- the general financial management of the EC funding, i.e. transmission pre-financing to all beneficiaries;
- providing to partners detailed information on consortium organisation and FP7 financial management rules during the Kick-off meeting;
- collecting and checking partner's financial information, submission of financial statements through NEF system;
- gathering all necessary information and materials for the preparation and submission of management report, activity report and deliverables.

WP leaders have had an important role for the preparation of project deliverables, acting as intermediary between beneficiaries and coordinator. They have also been responsible for the monitoring of WP activities and the quality check of deliverables and reports

Consortium Agreement

A Consortium Agreement regulating in particular the Intellectual Property Rights and Access Rights, the roles, functioning and responsibilities of consortium governing bodies, the liability of the partners and others, was signed by all CerCo partners in January 2010, before the start date of the project.

Communication Management

Due to the simple structure and composition of the consortium the communication strategy during the first 12 months required rather simple tools for communication in order to guarantee the exchange of information and data between partners, such as simple e-mail communication, website and consortium meetings.

Organisation of Kick-off and Periodical Meetings

The organisation of meetings was one key aspect for both coordination of future research activities and exchange of information between partners. All partners agreed that it is necessary and helpful to have direct contact between the involved research units especially for the purpose of a better integration among units. The project meetings allowed WP leaders and coordinator to monitor the project activities, prepare deliverables and plan future steps for the implementation of tasks.

During the first 12 months of CerCo we have organised 4 project meetings:

- **Kick-off meeting**, **Bologna, Italy, February 2010**. This first meeting concentrated both on coordination of research activities, planning of future work and financial and administrative organisation of the consortium presented by the Project Manager within UNIBO.

- **Project meeting**, **Edinburgh, Scotland, July 2010**. This meeting was scheduled at the end of the first six months of the project, just after the production of the important deliverable D2.1 (Compiler design and intermediate languages). Having taken the major design decisions, the project was aimed at fixing a more precise scheduling for the activities to be carried out in the remaining months. The meeting has also been scheduled to be co-located with FLOC, the Federated Logic conference, which is the largest venue of logicians. This provided us with an opportunity for informal dissemination of project results at the conference. All beneficiaries were represented.
- **Project meeting**, **Paris, France, September 2010**. The aim of this meeting was mainly to discuss the design of the costing compiler (D2.2) and the memory model and semantics of C (D4.1). In particular, the adoption of the MCS-51 microprocessor suggested to consider in D3.1 some ad-hoc extensions to the C language and required to take in account a non uniform memory model. At the meeting we discussed about the possibility of affecting D2.2 to support these features. Moreover, the choice of memory model will have a direct impact on WP4 too since the front-end and part of the back-end of the compiler will share the same memory model. We also discussed the possibility of having an implementer's meeting at the end of the period to address all the integration problems raised between the two meetings. All beneficiaries were represented.
- **Implementer's meeting, Bologna, Italy, January 2011**. This meeting was scheduled at the end of the tasks active during the first year. The objective of the meeting was to finalize D2.2 by solving the remaining integration problems with D3.1 and D4.1. The one-week-long meeting was used to put in close contact implementors from each project site in order to discuss the integration problems that required a tighter and promptly collaboration. Some of the problems were totally solved during the week. For the remaining one, we decided an integration strategy to be carried out in Task 2.4 during the next period of CerCo. All beneficiaries were represented.

Changes in the consortium

No major changes have occurred with regard to the organisation and management structure of the consortium. The only variation regards the change of the site leader of partner 3 UEDIN. Ian Stark, senior lecturer of The School of Informatics in the University of Edinburgh has replaced Randy Pollack as UEDIN site leader.

The legal statuses of the beneficiaries have remained unchanged.

Project planning and status

The project has followed the project plan (both technical and management) described in the Grant Agreement. Deliverables D6.1 (Project Web Site and Software Repository) was submitted on month 4 (+ 15 days) instead of month 3 because of problems with the delivery of the new dedicated server for CerCo. The Project Officer was informed of the problems and of the delay in the submission of the deliverable. As a consequence, also deliverables D2.1 and D6.2 were submitted on month 7 (+ 10 days) instead of month 6. The initial delay did not propagate any further. Indeed deliverables D3.1 and D4.1 were submitted on month 10 (+ 16 days), and

deliverable D2.2 on month 12 (+16 days) as expected. Deliverable D1.1 (Periodic Activity Report and Financial Statements) will be submitted within 60 days after end of the period.

3.5.2 Dissemination and use of the knowledge

Development of the Project website

In order to ease the creation of a community around the CerCo prototype, it was decided to prefer a Wiki-centred solution to a traditional Web Site. Our choice has been to install a personalised version of the Integrated SCM & Project Management tool Trac, accessible at http://cerco.cs.unibo.it. Trac provides in an integrated and lightweight solution:

1. A Wiki for the advertisement and dissemination of the project results, and for fostering discussions and contributions from the user community.
2. An issue tracking system that subsumes a bug tracking system and provides a basic control over the project advancement.
3. An on-line, simplied Web interface to the software repository.
4. A flexible system of plug-ins to add new functionalities or modify the behaviour of the basic ones.

The website acknowledges the funding received from the European Commission through the 7th Framework Programme and displays FP7 and FET logos.

Use of foreground and dissemination activities during this period

List of presentations given to international conferences:

- **Certified Complexity**. talk given by Claudio Sacerdoti Coen at Types 2010, Warsaw (15 October 2010).
- **Realizability Models for Cost-Preserving Compiler Correctness**, talk to be given by Marco Gaboardi at DICE 2010.
- **A canonical locally nameless formalisation of an algebraic logical framework**, talk given by Wilmer Ricciotti at Types 2010, Warsaw (13 October 2010).

List of seminars given to inviting institutions:

- **Certifying cost annotations in compilers**, talk given by Nicolas Ayache at the LIP6 laboratory of the University Paris 6 (18 November 2010);
- **A canonically nameless representation of binders**, talk given by Randy Pollack at U. Penn at Philadelphia (11 June 2010);
- **A canonically nameless representations of binders**, talk given by Randy Pollack at Lab PPS (Paris) to the INRIA-Rocquencourt team π r2 (26 November 2010);
- **Certifying cost annotations in compilers**, talk given by Roberto Amadio at ENS/Lyon.

Other seminars:

- **Operational Semantics in Specifications**, talk given by Brian Campbell at the Laboratory for Foundations of Computer Science, University of Edinburgh (16 March 2010).

List of papers currently submitted to international conferences or journals:

- **An executable formalisation of the MCS-51 microprocessor in Matita**, Dominic Mulligan and Claudio Sacerdoti Coen. Submitted to Interactive Theorem Proving 2011.
- **Certified complexity**, Dominic Mulligan and Claudio Sacerdoti Coen. Poster submission at FET11.
- **The Matita Interactive Theorem Prover,** Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen and Enrico Tassi. System description submitted to CADE 2011.
- **Certifying cost annotations in compilers**, Roberto Amadio, Nicolas Ayache, Yann Regis-Gianas, Ronan Saillard. Submitted to Formal Methods 2011.
- **Elementary affine lambda-calculus with multithreading and side effects**, A. Madet, R. Amadio. Submitted to TLCA 2011.
- **A decompilation of the pi-calculus and its applications to termination**, Roberto Amadio. Submitted to ICALP 2011.

3.6 Explanation of the use of the resources

3.6.1 Justification of major cost items and resources

The table below illustrate the resources used by each beneficiary, WP reference and justification.

Table 3.1 Personnel, subcontracting and other major Direct cost items for Beneficiary n. 1 **UNIBO** for period 1 (01/02/2010-31/01/2011)

Work Package	Item description	Amount	Explanations
1, 2,3,4,6	Personnel costs	71.721 €	The costs claimed under this category include: - 12 pm for permanent personnel (1 full professor, 1 assistant professor and 1 project manager) - 4 pm for 1 collaboration-professional contracts for research
2, 3, 4, 6	Travel and subsistance	3.435 €	Costs refer to the participation to project meetings and participation to conferences for dissemination of CerCo.
2, 3, 4	Equipments	1.014 €	Costs refer to the depreciated value of the network server. The server is dedicated to the CerCo project for dissemination (hosting the project Web site) and communication between partners (hosting the mailing list server and the message archive, but also acting as the central repository for the distributed versioning system used by partners to share and maintain the history of documents and prototypes).
2, 3, 4, 6	Other direct costs	3.308 €	Costs refer to organisation of kick-off meeting and participation to conferences for dissemination of CerCo (conference fee).
TOTAL DIRECT COSTS		79.478 €	

Table 3.2 Personnel, subcontracting and other major Direct cost items for Beneficiary n. 2 **UPD** for period 1 (01/02/2010-31/01/2011)			
Work Package	Item description	Amount	Explanations
1, 2	Personnel costs	105.765 €	The costs claimed under this category include: -5,6 pm for permanent personnel (1 full professor and 1 assistant; professor); -10,7 pm for 1 postdoctoral researcher and - 5,3 pm of 1 PhD student
2	Travel and subsistance	2.574 €	Participation CerCo meetings
2	consumables	2.647 €	electronic material and books
2	Other direct costs	2.085 €	Internship Kayvan Memarian
TOTAL DIRECT COSTS		113.071 €	

Table 3.3 Personnel, subcontracting and other major Direct cost items for Beneficiary n. 3 **UEDIN** for period 1 (01/02/2010-31/01/2011)			
Work Package	Item description	Amount	Explanations
2, 3	Personnel costs	41.934 €	Salary of postdoctoral research and 20% salary of site leader
2, 3	Travel costs	3.912 €	Costs refer to the participation to project meetings and participation to conferences for dissemination of CerCo.
2, 3	Other direct costs	1.484 €	Expenses for hosting meetings, small equipments and conference fees.
TOTAL DIRECT COSTS		47.330 €	

Budgeted versus Actual Costs

The tabular overview of budgeted costs and actual costs, by beneficiary and by major cost item including personnel, are reported in the table hereafter. The budgeted costs are taken from the Annex I of the Grant Agreement.

TABLE 4: COST/BUDGET FOLLOW-UP TABLE										
Contract N°:	**243381**	**Acronym: CerCo**				**Date:**	**20/03/2011**			
PARTI-CIPANTS	**TYPE of EXPENDITURE (as defined by participants)**	**BUDGET**	**ACTUAL COSTS (EUR)**				**Pct. spent**			**Remaining Budget (EUR)**
			Period 1	**P 2**	**P 3**	**Total**	**Year 1**	**Year 2**	**Total**	
		e	a1	b1	c1	e1	a1/e	a1+b1/e	a1+b1+c1/e	e-e1
Part. 1 UNIBO	**Total Person-month**	**79**	16			**16**	20%	20%	20%	**63,00**
	Personnel costs	**331.800**	71.721			**71.721**	22%	22%	22%	**260.079**
	Equipment	**4.000**	1.014			**1.014**	25%	25%	25%	**2.986**
	T&S	**30.000**	3.435			**3.435**	11%	11%	11%	**26.565**
	Other costs ('the rest')	**16.800**	3.308			**3.308**	20%	20%	20%	**13.492**
	Indirect costs	**225.960**	47.686			**47.686**	21%	21%	21%	**178.274**
	Total Costs	**608.560**	127.164	0,00	0,00	**127.164**	**21%**	**21%**	**21%**	**481.396**
Part. 2 UPD	**Total Person-month**	**66**	22			**22**	33%	33%	33%	**44,40**
	Personnel costs	**224.401**	105.765			**105.765**	47%	47%	47%	**118.636**
	Equipment	**0**	0			**0**				**0,00**
	T&S	**26.500**	2.573			**2.573**	10%	10%	10%	**23.927**
	Other costs ('the rest')	**10.014**	4.733			**4.733**	47%	47%	47%	**5.281**
	Indirect costs	**156.548**	67.842			**67.842**	43%	43%	43%	**88.706**
	Total Costs	**417.463**	180.913	0,00	0,00	**180.913**	**43%**	**43%**	**43%**	**236.550**
Part. 3 UEDIN	**Total Person-month**	**69**	10			**10**	15%	15%	15%	**58,92**
	Personnel costs	**310.617**	41.934			**41.934**	14%	14%	14%	**268.683**
	Equipment	**0**	0			**0**				**0,00**
	T&S	**27.011**	3.912			**3.912**	14%	14%	14%	**23.099**
	Other costs ('the rest')	**15.993**	1.484			**1.484**	9%	9%	9%	**14.509**
	Indirect costs	**144.099**	39.530			**39.530**	27%	27%	27%	**104.569**
	Total Costs	**497.720**	86.860	0,00	0,00	**86.860**	**17%**	**17%**	**17%**	**410.860**
TOTAL	**Total Person-month**	**214**	48			**48**	22%	22%	22%	**166,32**
	Personnel costs	**866.818**	219.420			**219.420**	25%	25%	25%	**647.398**
	Equipment	**4.000**	1.014			**1.014**	25%	25%	25%	**2.986**
	T&S	**83.511**	9.920			**9.920**	12%	12%	12%	**73.591**
	Other costs ('the rest')	**42.807**	9.525			**9.525**	22%	22%	22%	**33.282**
	Indirect costs	**526.607**	155.058			**155.058**	29%	29%	29%	**371.549**
	Total Costs	**1.523.743**	394.937			**394.937**	**26%**	**26%**	**26%**	**1.128.806**

3.6.2 Planned versus Actual effort

The following table illustrates the planned person-months as indicated in Annex I of the Grant Agreement and the actual effort consumption during the first reporting period.

TABLE 5: PERSON-MONTHS STATUS TABLE					
Grant Agreement N°: 243381		**TOTALS**	Coord. UNIBO	Partic. 2 UPD	Partic. 3 UEDIN
ACRONYM: CerCo					
PERIOD: 1/02/2010 - 31/01/2011					
Workpackage 1: Project Management	Actual WP total:	**2,39**	2	0,39	0
	Planned WP total:	**11**	9	1	1
Workpackage 2: Compiler Prototype (untrusted)	Actual WP total:	**30,28**	5	21,2	4,08
	Planned WP total:	**45**	7	30	8
Workpackage 3: Verified Compiler - front end	Actual WP total:	**7**	1	0	6
	Planned WP total:	**57**	3	3	51
Workpackage 4: Verified Compiler - back end	Actual WP total:	**6**	6	0	0
	Planned WP total:	**53**	45	6	2
Workpackage 5: Interfaces and interactive components	Actual WP total:	**0**	0	0	0
	Planned WP total:	**37**	9	24	4
Workpackage 6: Dissemination and exploitation	Actual WP total:	**2**	2	0	0
	Planned WP total:	**11**	6	2	3
Total Project Person-month	**Actual total:**	**45,67**	**14**	**21,6**	**10,1**
	Planned total:	**214**	**79**	**66**	**69**

3.6.3 Financial statements – Form C and Summary financial report - DRAFT

The Financial statements for all beneficiaries (Form Cs) , together with the Summary financial report will be submitted electronically using the NEF web tool.

3.6.4 Certificates on the financial statements

None of the beneficiaries has reached the expenditure threshold required for a certificate in period 1 of the project, thus no certificates have been provided from the CerCo beneficiaries in this reporting period.