Certified Complexity (CerCo)
Overall Presentation and Project Management

Claudio Sacerdoti Coen

May 16, 2013

SEVENTH FRAMEWORK PROGRAMME

Outline

Outline

SEVENTH FRAMEWORK PROGRAMME

The ICT of the future: Programs

We envision a (long term) future where formal methods will be pervasive:

Most programs will be (partially) specified and certified

The very same definition of "program" will change:

A program will be a triple made of a specification, an implementation and a proof certificate

Cfr: Proof Carrying Code (PCC)

The ICT of the future: the ITP point of view

There is always a trade off between expressivity (precision) and automation (partiality, abstraction).

Interactive theorem proving perspective:

- Maximal expressivity (precision) in specification
- Interactive proofs
- Abstraction, loss of precision as special cases of specification
- Automation as special cases of interaction

The ICT of the future: Compilation

In our long-term vision:

A compiler maps triples specification-implementation-certificate to triples specification-implementation-certificate.

What is investigated in CerCo:

Inclusion of intensional (non functional) properties into the specification to enable high level but precise reasoning.

The ICT of the future: Difficulties

Non functional properties (time, space, energy) are affected by:

- The compilation process (which is not compositional)
  E.g. the cost of different high level occurrences of the same instruction is different.
  E.g. aggressive optimizations that change the control flow
- Hardware status (caches, pipelines, branch prediction unit)

Questions and trade-offs:

- How much precision can be achieved?
- How complex are precise cost models for advanced compilation strategies and/or modern hardware?
- How much complexity can be tamed by abstraction and automation at the source code level?

The ICT of the future: Coping with the difficulties

In the project timeframe:

We started with a simple scenario:

- Conventional compilation model without changes to the control flow
- Deterministic widely deployed hardware model (no caching, pipelining)

We provided a formal certification of preservation of the non functional properties, modulo functional correctness

We have started to scale to most complex scenarios (without certification).

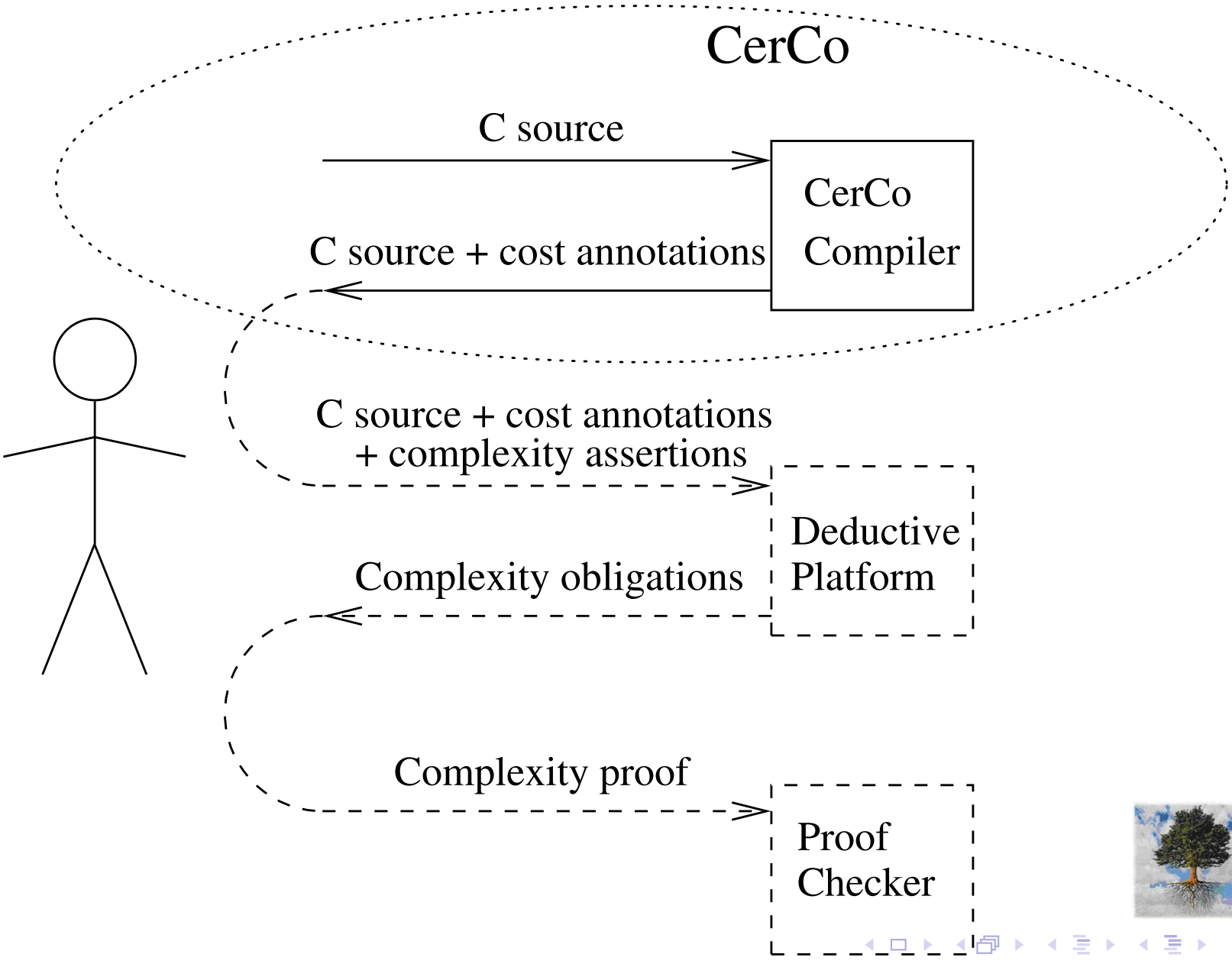
Short term objectives (achievements)

1. develop a compiler from ≈C to MCS-51 processor (1st period)
2. have the compiler infer from the assembly code a precise cost model for the source program (1st period)
3. certify the preservation of non functional properties by the compiler (3rd period, open proof obligations for functional properties)
4. embed the compiler in a prototypical environment for
   - stating cost invariants based on the inferred cost model
   - computing proof obligations
   - proving generated obligations automatically or interactively

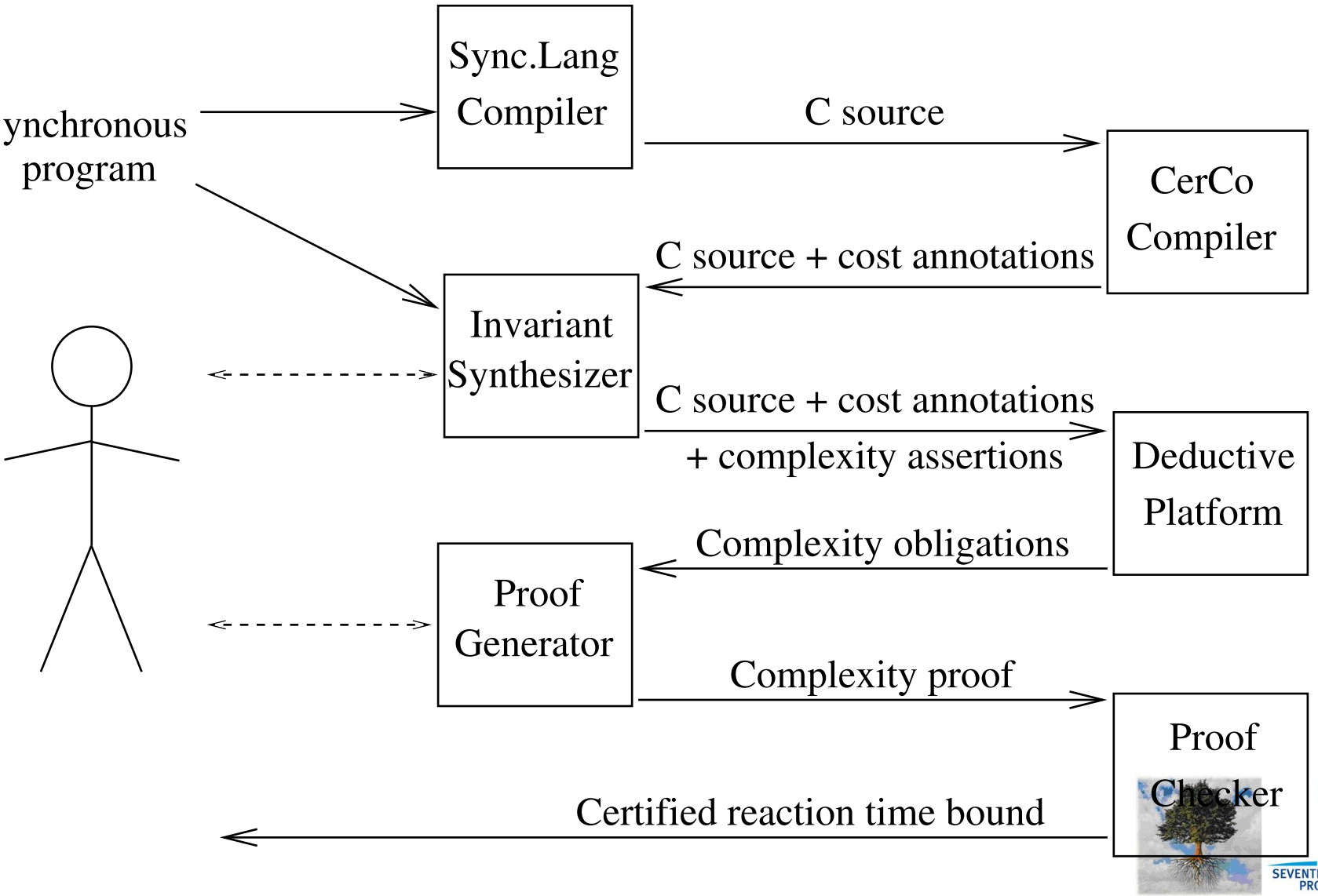
   (2nd period and 3rd period)
5. lift the technique to a synchronous language (2nd period)
6. Scale the approach to more complex scenarios (loop optimizations in 2nd period, investigations on modern hardware in 3rd period)

SEVENTH FRAMEWORK PROGRAMME

CerCo interaction diagram



Case study interaction diagram



Outline

Achievements (2nd period)

Main theoretical achievements (D2.1):

A methodology (basic labelling approach) for lifting in a provably correct way costs on the object code to costs in the source code

- Scaled to dependent labelling to cover optimizations that do not preserve the control flow (D5.1)
- Applied to imperative and functional languages (D5.1)
- The proof did not cover function calls

Achievements (2nd period)

Main practical achievements (1/2):

1. An untrusted compiler from C-Light to the MCS-51 processor that implements the methodology (D2.2), also extended with loop optimizations (D5.1)
2. Its formalization in Matita (D3.2, D4.2)
3. Formalized executable semantics for source, target and intermediate languages (D3.1, D3.3, D4.1, D4.3)

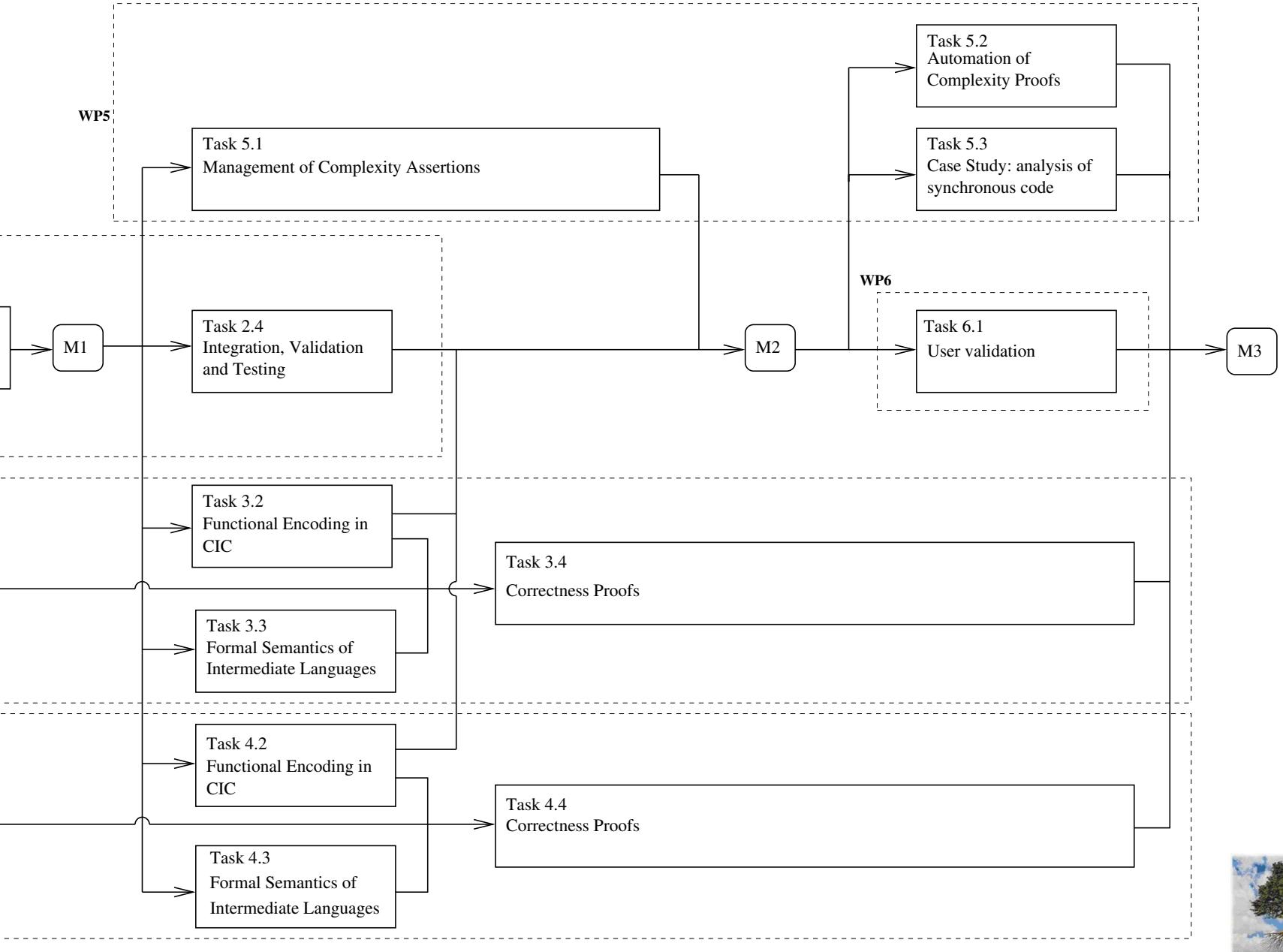
- The code was not extracted
- Not completely executable because of calls to untrusted code (cfr. translation validation, but also parsing/pretty printing)
- Code correct, not necessarily proof friendly

Achievements (1st period)

Main practical achievements (2/2):

Implementation for the Frama-C platform of a cost invariant synthesizer based on the CerCo's compiler (D5.1)

- Considered only simple cases
- Proof obligations produced by Jessie for the Why tool (Why3 in development, Frama-C under significant rewriting too)



SEVENTH FRAMEWORK PROGRAMME

Recommendations from the reviewers

"The consortium should concentrate on positioning the project with respect to the state of the art which requires a detailed literature survey. In particular, the consortium should clearly formulate what distinguishes the project from existing work and which results are clear improvements."

- We got in touch with other EU projects active in the area of WCET, in particular EMBounded, PROARTIS, TACLe, PARMERASA, T-CREST, REMS.
- Direct contacts during the CerCo events
- It really helped in understanding what are the relevant novelties, the possible application scenarios and the weaknesses from the engineering perspective
- More details in D1.4 and the final presentation

Recommendations from the reviewers

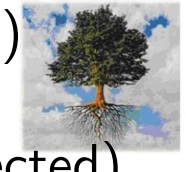
"The consortium should as soon as possible start with the verification of the compiler." "The reviewers are worried that the whole verification tasks cannot be successfully completed by the end of the third project period."

- The formal proof for the While language (1st period) was misleading: function calls introduce a much larger set of novel intensional invariants
- Moreover, the invariants are no longer expressed in terms of static code $\Rightarrow$ complete redesign of the proof strategy
- Had to consider stack space too
- Contingency plan: focus on the novel parts (preservation of non functional properties), also exploiting the uniform representation of backend languages
- Proof of preservation of costs completed up to proofs of preservation of functional properties

Recommendations from the reviewers

"The consortium should develop a code extractor for MATITA which allows obtaining executable code from specifications s.t. an executable version of the trusted CerCo compiler can be automatically generated. This extraction will be crucial to apply the CerCo approach to practical case studies and to present it to industry."

- Done
- Many issues faced on the quality of the extracted code (e.g. because of dependent types) (D5.2)
- Very nice example of very weakly typed extracted code: opens several nice research directions
  - Targeting OCaml's new first class modules to capture Σ-types
  - Targeting GADTs to capture most dependent types
  - A new technique to represent polymorphic variants and extensible records (had to be presented at TYPES'13)
  - Targeting GHC F_ω-like language (working prototype, does not accept all programs, much harder than expected)



Recommendations from the reviewers

"The consortium should concentrate on further developing and publishing the dependent labelling approach. In particular, the approach should be extended to further compiler optimizations, apart from loop peeling and hoisting, and be applied to more modern processor architectures, eventually aiming at multi-level caches and pipelines."

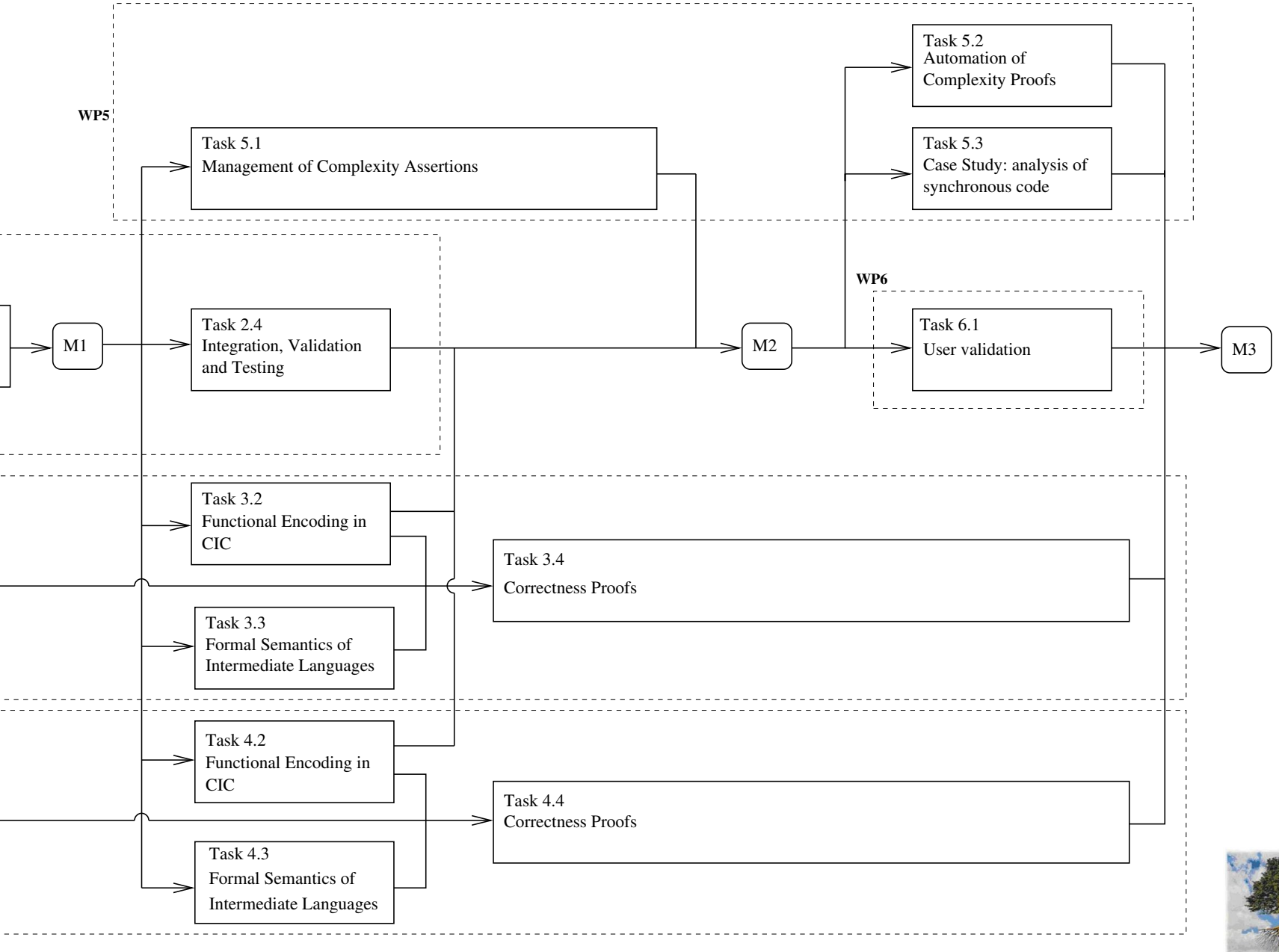
- Dependent labelling published at QAPL (journal version in preparation)
- One Post-Doc hired by UNIBO to work on application of dependent labelling to history sensitive hardware components (pipelines, caches)
- Outcome in my next talk

Recommendations from the reviewers

The reviewers required a revision of some deliverables and a few addenda.

The new versions and addenda were submitted to the EU as expected.

Outline



SEVENTH FRAMEWORK PROGRAMME

Achievements (3rd period) (WP3/WP4 talks)

From the formalized CerCo compiler (2nd period)

- untrusted code
- not optimized for proving
- with axiomatized fixpoint computer and graph colouring algorithm
- with formalized semantics for intermediate languages

to the trusted CerCo compiler (3rd period) (WP5 talk)

- extracted code integrated with untrusted code from the untrusted prototype
- preservation of costs (time + stack space) certified modulo preservation of functional properties

Achievements (3rd period) (WP5 talk)

From the untrusted CerCo prototype (2nd period)

- untrusted compiler +
  plug-in for Frama-C + Jessie + Why

to the trusted CerCo prototype (3rd period)

- trusted/untrusted compiler +
  plug-in for experimental Frama-C + Jessie + Why3
- code simplification, better coverage, integration with separation logic to cover arrays
- fine tuning of the generated invariants to match the automatic provers expertises
- a simplified interface for casual users
- testing
- Debian packages + Live CD for dissemination

Achievements (3rd period) (second WP4 talk)

Enlarging the scope of CerCo techniques

- application of the dependent labelling approach to cover certain cost models that capture parametricity of the cost on the internal state of hardware components that is history dependent
- sufficient to deal with pipelines (further research required)
- caches still problematic: towards probabilistic static analysis?

Other activities (3rd period) (WP6 talk)

Organization of CerCo events targeted to potential industrial stakeholders and the academic community.

Assessment

Assessment level and strategies:

1. Trusted CerCo compiler encoding: (D3.4/D4.4):
   - extracted code tested;
   - partially certified using Matita
2. Trusted CerCo prototype (D5.2, D6.3, M3):
   - tested (methodology + code) in the case study (D5.3);
   - tested on library of cryptographic functions (D6.3)
3. Case study: analysis of synchronous code (D5.3):
   - self testing code checks correctness and precision
   - tested on examples

Deviations from DoW

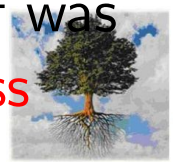
Minor deviations from the DoW (no significant impact on the schedule):

1. At month 28: submission of the addenda required by the reviewers. Done
2. Late delivery of D5.3 (at end of project, planned at 36). Most job completed during the 2nd period, we needed to test it again on the last version of the trusted compiler.
3. Late delivery of D6.6 (at end of project, planned at 33). The source code of the Debian packages and the infrastructure for the Live CD were ready on time, but the extracted code came later and was changing because of adjustments to the proofs.

Deviations from DoW

Major deviations from the DoW

1. DoW amended for administrative issues and to extend the length of the projects by two months
2. Research on handling history sensitive hardware in response to reviewers.
   No impact on the schedule, work done by additional Post-Doc.
3. Bring forward work on D5.3 from the third to the second period.
   Enabled by replacement of PhD student by Post-Doc in Paris
4. Compared to the formal proof for the Why language, the one for C revealed a much larger set of intensional invariants to be preserved; our contingency plan dictated that we focused on the aspects that are novel of our approach. The proof was finally completed up to proofs of functional correctness already covered by existing projects.



Outline

Management activities

Management WP1:

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

Financial management and coordination

During second period:

1. amend the DoW for administrative reasons (organizational changes in UNIBO) and to extend the duration to 38 months
2. transmission pre-financing to all beneficiaries
3. collecting and checking partner's financial information for the Second Progress Report
4. submission of financial statement through NEF system (in progress)
5. gathering all necessary information and submission of management reports, activity reports and deliverables
6. two sites (UNIBO and UEDIN) are undergoing the financial auditing

Periodic Meetings

Two project meetings and two short meetings during 3rd period (one planned in DoW):

- Short project meeting, Paris, March 2012.
- Short project meeting, Tallin, March 2012.
  At ETAPS, negotiation of the CerCo event at next ETAPS
- Project meeting, Bologna, May 2012
- Implementer's meeting, Edinburgh, August 2012