FICHE DE SYNTHESE DU PROJET « FOST »

Dossier de demande de labellisation du projet par le pôle

Dossier de soumission au GT Sécurité & Défense

Leader du projet

- Entité : INRIA / Centre de Recherche Saclay – Île-de-France
- Nom du contact : Sylvie Boldo
- Coordonnées (mail + téléphone) : sylvie.boldo@inria.fr
  01 74 85 42 26

Nom de l'appel à projet ANR visé : Programme Blanc

Partenaires du projet : Laboratoires publics, organismes de formation

Nom du laboratoire/ Etablissement/ Université	Nom du département	Nom de l'équipe de recherche	Adresse postale	Nom+mail+téléphone du Contact
INRIA	Centre de Recherche de Saclay – Île-de-France	PROVAL	Parc Orsay Université 4 rue Jacques Monod 91893 Orsay Cedex	Sylvie BOLDO *Sylvie.boldo@inria.fr* 01 74 85 42 26
INRIA	Centre de Recherche Paris - Rocquencourt	ESTIME	Domaine de Voluceau-Rocquencourt B.P. 105 78153 Le Chesnay Cedex	François CLÉMENT *Francois.Clement@inria.fr* *01 39 63 58 46*
CNRS Université Paris 13	LIPN	LCR	Institut Galilée – Université Paris-Nord 99, avenue Jean-Baptiste Clément 93430 Villetaneuse	Micaela Mayero *Micaela.Mayero@lipn.univ-paris13.fr* 01 49 40 36 91

Descriptif résumé du projet

FOST aims at developing and applying methods to formally prove the soundness of programs used in numerical analysis. In particular, we are interested in programs which often appear in the resolution of critical problems and in increasing their safety level. Many critical programs come from numerical analysis, but few people have ever tried to apply formal methods to this kind of programs. One reason is that formal methods were too immature to handle such problems. Formal method tools and in particular formal proof systems are becoming mature and are now able to deal with the real numbers and with floating-point numbers, which makes it possible to apply these systems to numerical analysis programs. Another main reason is that the communities are apart and seldom talk one to another. The numerical analysis community is used to prove with pen and paper that their method error is bounded. They usually discard the floating-point error as it is very different from what they are used to bound. They either do not know about or are afraid of formal method, which are considered too far away their usage. The formal methods community is now able to handle real numbers and mathematical properties. Proving these properties is often tedious and hardly rewarding. Therefore, numerical analysis is discarded. Moreover, FOST aims at providing reusable methods that are understandable by non-specialists of formal methods.
FOST is the association of several members with varied and appropriate competences (numericians, experts in proof assistant design, in proofs of programs, in real and floating-point numbers) and with a significant experience in their respective domains.

Organisation en sous-projets

1. Gradient

When coefficients in a partial differential equation are unknown, they can be estimated from some measurement of the solution. This parameter estimation problem is usually put in terms of the minimization of a least square function. The joint state technique is an efficient tool for computing the analytic gradient of this function. The singular value decomposition is a powerful tool for the deterministic sensitivity analysis. It can quantify the number of parameters that can be estimated from the solution measurements. This allows us to choose a ‘parametrization’ for the sought coefficients, and also to create measurement experiments. These techniques are for example used in seismic in order to probe the interior of Earth with pulse emissions. In this exact case, this consists in minimizing the function $J(p) = ½ \|d – F(p)\|^2$ where d stands for the experimental measure and F(p) the theoretic function. In order to compute this minimization, we find all the values such as grad(J(p))=0, where grad is the gradient. The gradient is a typical numerical analysis program we intend to formally prove, by bounding both the method error and the floating-point error.

2. Generalization

This task is subdivided into two sub-tasks as we are interested in bounding both the method error and the floating-point error:
- We want to deduce generic methods and results that can be applied to a large class of problems. For example, we will increase the order of the numerical scheme, or increase the space dimension or even consider other equations (elastic waves, Maxwell equations...). We will also address the simpler cases of Ordinary Differential Equations (ODE) and in particular study how our generic results could be applied to the classical 4th-order Runge-Kutta method (RK4). FOST will show the limits of CerPAN's approach and what genericity can be obtained from the already developed methods.
- The generalization on the floating-point error is based on the analytical error. FOST will determine if this technique can be applied to a large class of programs and under which conditions. Moreover, it will try to help to find out the analytical expression (automatically or not). The existence of a simple analytical expression is also under question: the convergence of the numeric scheme may be a sufficient condition for a convenient expression of the analytical error.

3. Toolbox

This task represents several ways to make formal methods accessible to non-specialists. The computing scientists are indeed not familiar with formal tools, and some pedagogy is needed to help them towards formal guarantees.

This toolbox can be thought as a comprehensive manual. It contains some documentation for annotating programs, methods to try to prove them and several real-life examples that are fully proved and explained step by step. We do not hope to convert computing scientists to formal methods. A first step would be a simple specification of their program that would bring more coherence. A part of the annotations could be proved using the Gappa tool that produces a formal proof by using interval arithmetic methods. Nevertheless, as soon as loops are involved, Gappa is usually not powerful enough to give interesting results. Nevertheless, partial formal proofs could be done automatically (such as array access or floating-point overflow).

Tâche/Tasks	Partenaires/Partners			Année 1 Year 1		Année 2 Year 2		Année 3 Year 3	
	1	2	3	6	12	18	24	30	36
Task 1.1 Gradient – method error									
Task 1.2 Gradient – floating-point error									
Task 2.1 Generalization – method error									
Task 2.2 Generalization – floating-point error									
Task 3 Toolbox									
Rapports d'avancement semestriel Progress report/expenses				☺	☺	☺	☺	☺	☺
Rapport final /Final report									☆

☺ : Rapport d'avancement semestriel/6 month-progress report

☆ : Rapport de synthèse et récapitulatif des dépenses/Final report and expenses summary

Principaux verrous technologiques à lever

The issue here is to study numerical analysis programs. Their particularity is that the method error is usually well-known and proved in pen-and-paper proofs. Getting a formal proof of correctness requires several steps:

- many mathematical properties have to be formalized and proved, e.g. Fourier transform. These properties are well-known, but their formalization may be long and tedious, while hardly rewarding.
- the proof of the mathematical error relies on the previous libraries. It may reveal gaps in the proof and will probably have to go into the very depths of the pen-and-paper proof.
- the floating-point proof is unknown: this aspect is usually discarded. Nevertheless, for this aspect, there is already a formal library of known facts about floating-point arithmetic that FOST can reuse.

The originality of FOST mainly lies in the meeting of two separated communities:

- computing scientists use computers to provide an answer and bound their error with pen-and-paper proofs. They do not go into formal proof as it is considered too complex. They usually discard floating-point errors.
- Formal methods provide the correctness of systems. Formal proofs may concern mathematical properties or correctness of programs. They do not go into numerical analysis as it is considered too complex and would require a large library of mathematical facts. This library would be long to develop and poorly rewarded.

The novelty in this approach is that we put together people from different fields in order to provide a high level of guarantee of real-life programs. FOST therefore needs people from different fields to work together. The scientific locks we plan to unfasten are:

- study a real program computing the gradient. This operator is well-known and often used. This is a good case study for our approach.
- study a large range of numerical programs. Some methods and results obtained previously can be applied (or not) to other programs. It is interesting to know which general methods we develop can be applied to a class of numerical applications.
- provide a toolbox for computing scientists. Formal methods are rather obscure for many mathematicians and we need a lot of pedagogy to bring them to formal tools.

FOST will also deal with two very different kinds of bounds: numerical programs are correct if both their method error and floating-point error are bounded. Therefore, we need to develop methods for bounding both errors. The methods will probably be very different and require expertise in two different domains.

- The method error mostly amongst to mathematical demonstrations. We will therefore need to formally prove many mathematical results and go into the very details of the pen-and-paper proofs. The consequence will be a library of mathematical facts.
- The floating-point error requires a deep study of the program. We will look into manual and automatic methods to be able to bound the final error due to roundings at each operation. The fact that these errors may compensate is to be taken into account for the final error to be reasonably small.

This double aspect of mathematical and computer arithmetic requirements is to be composed with the formal methods requirement of very detailed proofs. This is indeed ambitious as many competences are to be mixed in order to reach our goal.

Dissémination des résultats, retombées et attendus du projet

The expected results of FOST are:

- a gallery of formally proved numerical analysis programs;
- a library of mathematical facts to base upon in order to bound the method error;
- several generic methods and tools for formally proving numerical analysis programs;
  - for the method error;
  - for the floating-point error;
- a toolbox for computing scientists to add confidence in their programs.

Liens avec d'autres projets de SYSTEM@TIC PARIS-REGION

Some links will be established with the PFC project of System@TIC.

Le projet est-il soumis à d'autres pôles de compétitivité ? Si oui, lesquels ? Et dans quelles proportions ?

Non.

Durée prévue du projet (en mois) : 36

Date envisagée de début des travaux : Q1 2009

Coût du projet (en k€) :	276 k€
Coût de R&D en Ile de France (en k€) :	276 k€
Aide envisagée (en k€) :	112 k€