2021-03619 - PhD Position F/M [Cordi-S Campaign 2021 Grenoble] PhD on reversible semantics of programming languages and transactions

Type de contrat : CDD
Niveau de diplôme exigé : Bac + 5 ou équivalent
Autre diplôme apprécié : Master of Science in Computer Science or Mathematics
Fonction : Doctorant

A propos du centre ou de la direction fonctionnelle

Grenoble Rhône-Alpes Research Center groups together a few less than 650 people in 37 research teams and 8 research support departments.

Staff is localized on 5 campuses in Grenoble and Lyon, in close collaboration with labs, research and higher education institutions in Grenoble and Lyon, but also with the economic players in these areas.

Present in the fields of software, high-performance computing, Internet of things, image and data, but also simulation in oceanography and biology, it participates at the best level of international scientific achievements and collaborations in both Europe and the rest of the world.

Contexte et atouts du poste

The PhD work will take place as part of an ongoing collaboration between the SPADES team at INRIA Grenoble and the FOCUS of INRIA Sophia, at the University of Bologna.

Regular visits between Grenoble and Bologna are foreseen during the 3 years of the PhD work.

The PhD candidate will be co-advised by Pr Ivan Lanese from the FOCUS team in Bologna and Jean-Bernard Stefani from the SPADES team in Grenoble.

Mission confiée

Subject: reversible semantics of programming languages and transactional models

Reversible computing is a paradigm allowing programs to compute not only in the standard, forward direction, but also backwards, recovering past states. Reversible computing has attracted interest due to a number of relevant applications, including low-power computing, biological modelling, simulation, robotics and, particularly relevant here, debugging. In concurrent systems, the most used notion of reversibility is causal-consistent reversibility [3,4], which allows any action to be undone provided that its consequences, if any, are undone beforehand. Equivalently, independent actions can be undone in any order, while dependent actions need to be undone in reverse order. Most programs written in mainstream languages are not reversible, and even less causal-consistent reversible, since computation frequently causes loss of information and does not record causality information. However, one can try to extend a given programming language or formalism by tracking enough information to make it causal-consistent reversible, and this has been done for a number of formalisms, including CCS, the pi-calculus, P/T nets and the programming language Erlang. The work on Erlang lead to the development of CauDE [5,6], a causal-consistent reversible debugger for Erlang.

The PhD work proposed would seek to contribute to the theory of concurrent reversible programming languages and its application to the debugging of concurrent programs. Two main topics are envisaged:

(1) the automated derivation of a reversible semantics for concurrent languages;
(2) the study of transactional programming abstractions through the lens of reversibility.

Automated derivation of reversible semantics

Currently, the procedure to define a reversible extension of a given concurrent formal model or programming language is mostly ad hoc, and preliminary proposals for general techniques have started to appear only recently [1,2]. While [2] only discusses how to prove properties after a reversible extension has been defined, [1] proposes an automatic technique to define a causal-consistent reversible extension of a given forward model. However, [1] currently has a number of limitations, which need to be overcome to ensure applicability in real contexts. First, [1] assumes a simple notion of causality stating that two actions are dependent if one consumes a resource that the other produces. This is suitable for dependencies originated from message passing, but, e.g., not from shared memory. Second, [1] only discusses the uncontrolled reversibility, where no policy is provided to decide whether to go forwards or backwards, nor to select the specific action to execute or undo. Uncontrolled reversibility is good for foundational studies, but applications (such as debugging) require forms of control of reversibility as used, e.g., in a rollback operator such as the one of CauDE. Finally, the current approach is fully theoretical, and some refinement is needed to apply it in practice, e.g., to find finite representations for the current infinite sets of rules. This can lead to a direct application in form...
of a program that takes a description of a concurrent system in form of a Maude module, and produces a Maude module corresponding to its causal-consistent reversible extension. A more challenging implementation would be an extension of CauDEr so to make it parametric on the input language: this will allow one to cope with additional features of the language without having to modify CauDEr code, just by providing a specification of the new features.

Transactions via reversibility

Transactions are key abstractions for concurrent programming with access to a shared data store (transactional memory or transactions in database management systems), but they come in diverse forms and semantics. An intriguing possibility, considering that most transaction abstractions involve some form of rollback to undo the effect of failed transactions, is to understand transactions as high-level constructs built on a reversible semantics, much as the rollback operator in CauDEr can be defined on a reversible semantics of Erlang. Very preliminary work in this direction appeared in [7] and the transactional model considered in this paper only dealt with some form of communicating transactions. Much more work would be needed to (a) take into account access to a shared data store, under different consistency models, and (b) to take into account different isolation constraints between transactions. Although the work proposed here would be primarily theoretical, it could find a direct application in the debugging of concurrent programs with a shared memory and attendant memory model and transactional memory constructs, going beyond the message passing concurrency model handled by CauDEr.

References


The successful applicant will be expected to contribute to this research effort, which will include both semantic foundations, programming abstractions, and practical implementations.

Principales activités

Main activities (5 maximum):

- State of the art analysis in the following domains: reversible concurrent programming models and languages, transaction models and programming abstractions, transactional isolation and formalization
- Study of automated derivation of reversible semantics for concurrent languages
- Study of transactional programming abstractions by means of a reversible semantics
- Prototyping of extensions to the CauDEr reversible debugger
- Preparing submissions and participations to relevant journals and international conferences

Compétences

Technical skills and level required:

- Good background on programming languages and concurrency theory (Master Degree)
- Good programming expertise in a modern programming language such as Ocaml, Python, Erlang

Languages:

- English (spoken and written)
- knowledge of French or Italian welcome

Relational skills:

- team spirit
- oral and presentation skills

Other valued appreciated:

Avantages

- Subsidized meals
- Partial reimbursement of public transport costs
- Leave: 7 weeks of annual leave + 10 extra days off due to RTT (statutory reduction in working hours) + possibility of exceptional leave (sick children, moving home, etc.)
- Possibility of teleworking two days per week and flexible organization of working hours
- Professional equipment available (videoconferencing, loan of computer equipment, etc.)
- Social, cultural and sports events and activities
- Access to vocational training
- Social security coverage

Rémunération


Monthly salary after taxes: around 1596,05€ for 1st and 2nd year. 1678,99€ for 3rd year (medical insurance included, income tax excluded).