Offer #2025-09115

Doctorant F/H Vérification de la correction fonctionnelle de composants de systèmes d'exploitation

The offer description below is in French

Contract type : Fixed-term contract

Level of qualifications required : Graduate degree or equivalent

Fonction : PhD Position

Context

Operating system code (kernel, device drivers) has become increasingly complex, yet it still involves a

large amount of low level code which manipulates intricate data-structures. Moreover, the stability of the OS

(operating system) is critical for any computing system, due to its role managing the other processes and all the

resources of the machine (CPU, memory, I/O access). Therefore, it is important to check that OS code meets

important correctness properties, such as, in increasing order of difficulty (1) basic safety (absence of runtime

errors), (2) the preservation of structural invariants, and (3) more advanced functional properties (e.g., that a

scheduler correctly implements the intended scheduling algorithm).

Static analysis based on abstract interpretation aims at computing program invariants so as to discharge

the verification of such properties. It is automatic and sound (it never validates a program that is incorrect)

but incomplete, which means that it may fail to prove the correctness of a program that meets the property of

interest. Therefore, the practical design of static analyzers remains a grand challenge. In particular, the design of expressive and computable families of predicates requires a lot of effort.

In the case of OS code as described above, several families of static analysis have achieved impressive

progress. In particular, shape analysis relies on predicates which describe data-structures of unbounded size in

a compact way, which allows generalizing properties observed over short symbolic executions into expressive

program invariants, for many programs that manipulate complex data-structures. In particular, data-structures

like linked pointer structures (lists and trees), and linear structures (arrays) are well handled by existing shape

analysis techniques.

However, many important issues remain. In particular, OS code often mixes and overlays arrays and linked

structures. Such code often relies on structural patterns that are not so well-covered by existing shape analyses.

If we consider the combinations of arrays and linked-structures, several works (by P. Sotin, J. Liu and X. Rival

[4, 5]) have allowed to describe precisely structures that consist of an array that contains a linked list structure,

however, more complex combinations are not handled by any existing shape analysis. For instance, structures

made of an array the elements of which point to separate linked lists cannot be described by existing abstract

domains. Another poorly handled structural pattern consists of overlaid lists, where large structures are shared

in multiple lists [2].

In this thesis, we propose to design abstract domains for combinations of arrays and linked structures so

as to handle both aforementioned patterns. The design of abstract domains will include the formalization of

abstract elements, of the concretization function, and of the abstract operations. Moreover, we will implement

the proposed abstract domains into the MemCAD static analyzer [1, 3], which is developed in the ANTIQUE

INRIA Research Team, and we will evaluate the implementation based on existing OS code, and assess whether

it can achieve the proof of aforementioned properties (1), (2), and (3). Finally, we expect our study to also

produce generalization of abstract domains used in shape analysis.

Assignment

Operating system code (kernel, device drivers) has become increasingly complex, yet it still involves a

large amount of low level code which manipulates intricate data-structures. Moreover, the stability of the OS

(operating system) is critical for any computing system, due to its role managing the other processes and all the

resources of the machine (CPU, memory, I/O access). Therefore, it is important to check that OS code meets

important correctness properties, such as, in increasing order of difficulty (1) basic safety (absence of runtime

errors), (2) the preservation of structural invariants, and (3) more advanced functional properties (e.g., that a

scheduler correctly implements the intended scheduling algorithm).

Static analysis based on abstract interpretation aims at computing program invariants so as to discharge

the verification of such properties. It is automatic and sound (it never validates a program that is incorrect)

but incomplete, which means that it may fail to prove the correctness of a program that meets the property of

interest. Therefore, the practical design of static analyzers remains a grand challenge. In particular, the design

of expressive and computable families of predicates requires a lot of effort.

In the case of OS code as described above, several families of static analysis have achieved impressive

progress. In particular, shape analysis relies on predicates which describe data-structures of unbounded size in

a compact way, which allows generalizing properties observed over short symbolic executions into expressive program invariants, for many programs that manipulate complex data-structures. In particular, data-structures

like linked pointer structures (lists and trees), and linear structures (arrays) are well handled by existing shape

analysis techniques.

However, many important issues remain. In particular, OS code often mixes and overlays arrays and linked

structures. Such code often relies on structural patterns that are not so well-covered by existing shape analyses.

If we consider the combinations of arrays and linked-structures, several works (by P. Sotin, J. Liu and X. Rival

[4, 5]) have allowed to describe precisely structures that consist of an array that contains a linked list structure,

however, more complex combinations are not handled by any existing shape analysis. For instance, structures

made of an array the elements of which point to separate linked lists cannot be described by existing abstract

domains. Another poorly handled structural pattern consists of overlaid lists, where large structures are shared

in multiple lists [2].

In this thesis, we propose to design abstract domains for combinations of arrays and linked structures so

as to handle both aforementioned patterns. The design of abstract domains will include the formalization of

abstract elements, of the concretization function, and of the abstract operations. Moreover, we will implement

the proposed abstract domains into the MemCAD static analyzer [1, 3], which is developed in the ANTIQUE

INRIA Research Team, and we will evaluate the implementation based on existing OS code, and assess whether

it can achieve the proof of aforementioned properties (1), (2), and (3). Finally, we expect our study to also

produce generalization of abstract domains used in shape analysis.

Main activities

Operating system code (kernel, device drivers) has become increasingly complex, yet it still involves a

large amount of low level code which manipulates intricate data-structures. Moreover, the stability of the OS

(operating system) is critical for any computing system, due to its role managing the other processes and all the

resources of the machine (CPU, memory, I/O access). Therefore, it is important to check that OS code meets

important correctness properties, such as, in increasing order of difficulty (1) basic safety (absence of runtime

errors), (2) the preservation of structural invariants, and (3) more advanced functional properties (e.g., that a

scheduler correctly implements the intended scheduling algorithm).

Static analysis based on abstract interpretation aims at computing program invariants so as to discharge

the verification of such properties. It is automatic and sound (it never validates a program that is incorrect)

but incomplete, which means that it may fail to prove the correctness of a program that meets the property of

interest. Therefore, the practical design of static analyzers remains a grand challenge. In particular, the design

of expressive and computable families of predicates requires a lot of effort.

In the case of OS code as described above, several families of static analysis have achieved impressive

progress. In particular, shape analysis relies on predicates which describe data-structures of unbounded size in

a compact way, which allows generalizing properties observed over short symbolic executions into expressive

program invariants, for many programs that manipulate complex data-structures. In particular, data-structures

like linked pointer structures (lists and trees), and linear structures (arrays) are well handled by existing shape

analysis techniques.

However, many important issues remain. In particular, OS code often mixes and overlays arrays and linked structures. Such code often relies on structural patterns that are not so well-covered by existing shape analyses.

If we consider the combinations of arrays and linked-structures, several works (by P. Sotin, J. Liu and X. Rival

[4, 5]) have allowed to describe precisely structures that consist of an array that contains a linked list structure,

however, more complex combinations are not handled by any existing shape analysis. For instance, structures

made of an array the elements of which point to separate linked lists cannot be described by existing abstract

domains. Another poorly handled structural pattern consists of overlaid lists, where large structures are shared

in multiple lists [2].

In this thesis, we propose to design abstract domains for combinations of arrays and linked structures so

as to handle both aforementioned patterns. The design of abstract domains will include the formalization of

abstract elements, of the concretization function, and of the abstract operations. Moreover, we will implement

the proposed abstract domains into the MemCAD static analyzer [1, 3], which is developed in the ANTIQUE

INRIA Research Team, and we will evaluate the implementation based on existing OS code, and assess whether

it can achieve the proof of aforementioned properties (1), (2), and (3). Finally, we expect our study to also

produce generalization of abstract domains used in shape analysis.

Skills

Compétences techniques et niveau requis :

Langues :

Compétences relationnelles :

Compétences additionnelles appréciées :

Benefits package

- Restauration subventionnée

- Transports publics remboursés partiellement
- Congés: 7 semaines de congés annuels + 10 jours de RTT (base temps plein) + possibilité d'autorisations d'absence exceptionnelle (ex : enfants malades, déménagement)
- Possibilité de télétravail (après 6 mois d'ancienneté) et aménagement du temps de travail
- Équipements professionnels à disposition (visioconférence, prêts de matériels informatiques, etc.)
- Prestations sociales, culturelles et sportives (Association de gestion des œuvres sociales d'Inria)
- Accès à la formation professionnelle
- Sécurité sociale

General Information

- **Theme/Domain :** Proofs and Verification
  Software engineering (BAP E)
- **Town/city :** Paris
- **Inria Center :** Centre Inria de Paris
- **Starting date :** 2025-09-01
- **Duration of contract :** 3 years
- **Deadline to apply :** 2025-07-31

Contacts

- **Inria Team :** ANTIQUE
- **PhD Supervisor :**
  Rival Xavier / Xavier.Rival@inria.fr

About Inria

Inria is the French national research institute dedicated to digital science and technology. It employs 2,600 people. Its 200 agile project teams, generally run jointly with academic partners, include more than 3,500 scientists and engineers working to meet the challenges of digital technology, often at the interface with other disciplines. The Institute also employs numerous talents in over forty different professions. 900 research support staff contribute to the preparation and development of scientific and entrepreneurial projects that have a worldwide impact.

The keys to success

Vous pouvez donner là, un portrait à "gros traits" du (de la) collaborateur(trice) attendu(e) : ce que vous voyez comme nécessaire et suffisant et qui peut associer :

- goûts et appétences,
- domaine d'excellence,
- éléments de personnalité ou de caractère,
- savoir et savoir faire transversaux...

Cette rubrique permet de compléter et alléger (réduire) la liste plus formelle des compétences :

- "Se sentir à l'aise dans un environnement de dynamique scientifique, aimer apprendre et écouter sont des qualités essentielles pour réussir cette mission."
- " Passionné(e) par l'innovation, avec une expertise dans le développement Ruby on Rail et une grande capacité de conviction. Une thèse dans le domaine *** constitue un réel atout."

Warning : you must enter your e-mail address in order to save your application to Inria. Applications must be submitted online on the Inria website. Processing of applications sent from other channels is not guaranteed.

Instruction to apply

Defence Security :
This position is likely to be situated in a restricted area (ZRR), as defined in Decree No. 2011-1425 relating to the protection of national scientific and technical potential (PPST).Authorisation to enter an area is granted by the director of the unit, following a favourable Ministerial decision, as defined in the decree of 3 July 2012 relating to the PPST. An unfavourable Ministerial decision in respect of a position situated in a ZRR would result in the cancellation of the appointment.

Recruitment Policy :
As part of its diversity policy, all Inria positions are accessible to people with disabilities.