Offre n°2025-09115

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

Type de contrat : Fixed-term contract

Niveau de diplôme exigé : Graduate degree or equivalent

Fonction : PhD Position

Contexte et atouts du poste

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.

Mission confiée

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.

Principales activités

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.

Compétences

Compétences techniques et niveau requis :

Langues :

Compétences relationnelles :

Compétences additionnelles appréciées :

Avantages

- 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

Informations générales

- **Thème/Domaine :** Proofs and Verification
  Software engineering (BAP E)
- **Ville :** Paris
- **Centre Inria :** Centre Inria de Paris
- **Date de prise de fonction souhaitée :** 2025-09-01
- **Durée de contrat :** 3 years
- **Date limite pour postuler :** 2025-07-31

Contacts

- **Équipe Inria :** ANTIQUE
- **Directeur de thèse :**
  Rival Xavier / Xavier.Rival@inria.fr

A propos d'Inria

Inria est l'institut national de recherche dédié aux sciences et technologies du numérique. Il emploie 2600 personnes. Ses 215 équipes-projets agiles, en général communes avec des partenaires académiques, impliquent plus de 3900 scientifiques pour relever les défis du numérique, souvent à l'interface d'autres disciplines. L'institut fait appel à de nombreux talents dans plus d'une quarantaine de métiers différents. 900 personnels d'appui à la recherche et à l'innovation contribuent à faire émerger et grandir des projets scientifiques ou entrepreneuriaux qui impactent le monde. Inria travaille avec de nombreuses entreprises et a accompagné la création de plus de 200 start-up. L'institut s'e?orce ainsi de répondre aux enjeux de la transformation numérique de la science, de la société et de l'économie.

L'essentiel pour réussir

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."

Attention: Les candidatures doivent être déposées en ligne sur le site Inria. Le traitement des candidatures adressées par d'autres canaux n'est pas garanti.

Consignes pour postuler

Sécurité défense :
Ce poste est susceptible d'être affecté dans une zone à régime restrictif (ZRR), telle que définie dans le décret n°2011-1425 relatif à la protection du potentiel scientifique et technique de la nation (PPST). L'autorisation d'accès à une zone est délivrée par le chef d'établissement, après avis ministériel favorable, tel que défini dans l'arrêté du 03 juillet 2012, relatif à la PPST. Un avis ministériel défavorable pour un poste affecté dans une ZRR aurait pour conséquence l'annulation du recrutement.

Politique de recrutement :
Dans le cadre de sa politique diversité, tous les postes Inria sont accessibles aux personnes en situation de handicap.