Application BASTRI

Fiches Equipes

GALLINETTE (SR0841VR)

Gallinette : vers une nouvelle génération d'assistant à la preuve
GALLINETTE (SR0785LR) GALLINETTE

Statut: Décision signée

Responsable : Nicolas Tabareau

Mots-clés de "A - Thèmes de recherche en Sciences du numérique - 2024" : A2.1.1. Sémantique des langages de programmation , A2.1.2. Programmation impérative , A2.1.3. Programmation orientée objet , A2.1.4. Programmation fonctionnelle , A2.1.11. Langages de preuve , A2.2.3. Gestion mémoire , A2.4.3. Preuves , A7.2.3. Assistants de preuve , A7.2.4. Formalisation mécanisée des mathématiques , A8.4. Calcul formel, calcul algébrique

Mots-clés de "B - Autres sciences et domaines d'application - 2024" : B6.1. Industrie du logiciel

Domaine : Algorithmique, programmation, logiciels et architectures
Thème : Preuves et vérification

Période : 01/06/2018 -> 31/12/2028
Dates d'évaluation : 20/03/2019 ,

Etablissement(s) de rattachement : IMT ATLANTIQUE, NANTES U.
Laboratoire(s) partenaire(s) : LS2N (UMR6004)

CRI : Centre Inria de l'Université de Rennes
Localisation : Université de Nantes - LS2N
Code structure Inria : 031120-1

Numéro RNSR : 201722488Z
N° de structure Inria: SR0841VR

Présentation

L'EPI Gallinette vise à développer une nouvelle génération d'assistants à la preuve, avec la conviction que les expériences pratiques doivent aller de pair avec les investigations fondamentales:

- L'objectif est de faire progresser les assistants à la preuve à la fois en tant que langages de programmation certifiés et systèmes logiques mécanisés. La programmation avancée et les paradigmes mathématiques doivent être intégrés, notamment types dépendants et effets. L'approche distinctive consiste à implémenter une nouvelle programmation et des paradigmes logiques au-dessus de Coq en considérant cette dernière comme un langage cible pour la compilation.
- Le but des recherches plus fondamentales est d'étendre les limites de la correspondance de Curry-Howard. Elle est considérée à la fois comme une base pour les langages de programmation et la logique, et comme un fournisseur de techniques essentielles au développement des assistants de preuve. Dans cette perspective, le développement d'assistants à la preuve est vu comme une expérience totale utilisant la correspondance dans tous les aspects: les langages de programmation, la théorie des types, la théorie de la preuve, la réécriture et l'algèbre.

Axes de recherche

1. Améliorer la puissance informatique et logique des assistants de preuve

La démocratisation des assistants de preuve basée sur la théorie des types souffre d'un inconvénient majeur, l'inadéquation entre la conception de l'égalité en mathématiques et l'égalité en théorie des types. En effet, certains principes de base qui sont implicitement utilisés en mathématiques - tels que le principe d'extensionalité propositionnelle de Church, qui dit que deux propositions sont égales lorsqu'elles sont logiquement équivalentes - ne peuvent pas être déduits dans la théorie des types. Plus problématiquement du point de vue informatique, le concept de base de deux fonctions égales lorsqu'elles sont égales à chaque "point" de leur domaine n'est pas non plus dérivable et doit être défini comme un axiome. Bien sûr, ces principes sont compatibles avec la théorie des types et leur ajout en tant qu'axiomes est sûr. Mais tout

Contact

- **Responsable :** Nicolas Tabareau
- **Tél :** +3.3 .(0.)2. 5.1 .85. 8.2 .37
- **Secrétariat Tél :** +3.3 .(0.)2. 5.1 .85. 8.7 .24

En savoir plus

- Site de l'équipe
- Site sur inria.fr
- Site du responsable
- Derniers Rapports d'Activité : 2018 , 2019 , 2020 , 2021 , 2022 , 2023 , 2024

Documents sur la structure

- Intranet
- Privés

Décisions

- 12889 (12/06/2018) : création
- 14025 (16/12/2019) : prolongation
- 16659 (11/12/2023) : prolongation
- 16998 (26/04/2024) : prolongation
- 17520 (04/12/2024) : prolongation

Localisation

- **Adresse postale :** Université de Nantes 2 Chemin de la Houssinière 44300 Nantes France
- **Coordonnées GPS :** *Non renseignées*

développement les utilisant dans une définition produira un morceau de code qui ne calcule pas, étant bloqué aux points où les axiomes ont été utilisés, parce que les axiomes sont des boîtes noires computationnelles.

Nous proposons d'étudier comment des transformations logiques expressives (comme le forcing, la faisceautisation, ...) pourraient être utilisées pour améliorer le pouvoir de calcul et de logique des assistants de preuve - avec un angle particulier vers l'intégration à l'assistant de preuve Coq.). L'un des principaux aspects, en relation avec le projet ERC CoqHoTT, est l'intégration de nouveaux concepts autour de la théorie homotopique des types tels que l'univalence, ou des types inductifs supérieurs.

2. Effets dans la théorie des types

Nous proposons d'incorporer des effets dans la théorie des assistants de preuve. Nous remarquons que l'enjeu n'est pas seulement la certification de programmes avec des effets, mais qu'elle a aussi des implications en termes de sémantique et de logique.

La théorie des types est basée sur le lambda calcul qui est purement fonctionnel. Constatant que tout programme réaliste contient des effets (état, exceptions, entrées-sorties ...), de nombreux travaux se concentrent sur la programmation certifiée avec effets: notamment Ynot, et plus récemment F * et Idris, qui proposent différentes manières d'encapsuler les effets et de limiter la dépendance des types sur des termes efficaces. Les effets sont soit spécialisés, avec des monades avec des pré et post-conditions de style Hoare trouvées dans Ynot ou F *, ou plus généraux, avec des effets algébriques implémentés dans Idris. D'un autre côté, il y a un déficit d'investigations logiques et sémantiques.

Pourtant, une théorie des types qui intègre les effets aurait des implications logiques, algébriques et computationnelles à travers la correspondance de Curry-Howard. Par exemple, les effets tels que les opérateurs de contrôle délimités fournissent des interprétations computationnelles aux axiomes tels que A ¬¬A et ¬xA x¬A, contrairement à la théorie des types habituelle qui considère que la logique classique est non-constructive. Toute une littérature sur le contenu constructif des preuves classiques doit être explorée et intégrée.

L'objectif est de développer une théorie des types avec des effets qui rendent compte à la fois des expériences pratiques en programmation certifiée et des indices provenant de la sémantique dénotationnelle et des phénomènes logiques, à partir du cadre unificateur du call-by-push-value (linéaire).

3. Améliorer l'interconnexion entre les assistants de preuve et les langages de programmation courants

Nous proposons d'intérioriser les concepts trouvés dans les langages de programmation courants, comme les objets, au sein des assistants de preuve, afin de fournir une meilleure intégration avec la pratique dans l'industrie. Par internalisation, nous entendons une intégration superficielle d'une partie des langages de programmation courants afin de pouvoir certifier et synthétiser des programmes écrits dans ces langages directement à l'intérieur des assistants de preuve.

Mais il existe un fossé entre le paradigme de programmation sous-jacent à la correspondance de Curry-Howard, la programmation fonctionnelle pure et le paradigme de programmation dominant pour l'écriture d'applications logicielles dans l'industrie, notamment la programmation orientée objet. L'un des principaux problèmes avec l'approche de l'internalisation est que la théorie des types est contrainte par une discipline de type stricte qui est souvent plus flexible sinon absente dans les langages courants. Pour faire face à ce problème, l'EPI Gallinette mettra tout en œuvre pour développer le typage graduelle en théorie des types afin de permettre une intégration progressive du code issu d'un monde plus permissif. Aussi, pour permettre l'interconnexion, nous prévoyons de développer l'interopérabilité entre types dépendants présents dans la théorie des types et types simples avec des propriétés présents dans les langages dominants. Plus généralement, l'internalisation fournira un cadre commun pour exprimer une série d'oppositions:

- programmation orientée objet versus programmation fonctionnelle,
- polymorphisme de l'objet versus polymorphisme paramétrique,
- interopérabilité versus intra-opérabilité,
- abstraction de données procédurales par rapport aux types de données abstraits,
- sous-typage subsomptif versus sous-typage coercitif,
- interprétations co-algébriques versus algébriques.

Relations industrielles et internationales

ERC Consolidator Grant Fresco (https://fresco.gitlabpages.inria.fr/)

Nomadic Labs

Inria Associate Team with UChile