LTP 2013

Journée annuelle du groupe Langages, Types et Preuves du GDR GPL

LaBRI, Université de Bordeaux, 18 Novembre 2013

Résumés des Exposés

9h00–9h25

Modules over monads for operational semantics
Benedikt Ahrens, IRIT.

We define a notion of 2-signature, which allows to specify the types and terms of a (programming) language as well as reductions, i.e. directed rewrite rules, on those terms. To any such 2-signature we construct a category of models of that signature, such that the language generated by the signature - equiped with the specified reductions - is the initial such model. Initiality then yields an iteration operator which allows to specify translations which have good properties by construction - in particular, such translations are compatible with reduction and substitution in the source and target languages.

9h30–9h55

Program Transformation for Non-interference Verification on Programs with Pointers
Mounir Assaf, CEA.

Novel approaches for dynamic information flow monitoring are promising since they enable permissive (accepting a large subset of executions) yet sound (rejecting all unsecure executions) enforcement of non-interference. In this paper, we present a dynamic information flow monitor for a language supporting pointers. Our flow-sensitive monitor relies on prior static analysis in order to soundly enforce non-interference. We also propose a program transformation that preserves the behavior of initial programs and soundly inlines our security monitor. This program transformation enables both dynamic and static verification of non-interference.

10h00–10h25

Vérification de programmes avec lieurs
Martin Clochard, LRI.

Nous proposons une approche générique pour manipuler les types de données avec lieurs à la fois dans un programme et sa spécification d'une façon qui facilite le raisonnement automatique sur de tels types de données et qui conduit également à du code raisonnablement efficace. Notre méthode est implémentée

dans l'environnement de preuve de programmes Why3. Nous la validons sur les examples d'un interpréteur du lambda-calcul avec plusieurs stratégies de réduction et d'un petit prouveur automatique basé sur la méthode des tableaux.

10h30–11h00

Pause

11h00–11h25

Une étude de cas en spécification et implantation chaînée certifiée utilisant des orbites : les hypercartes combinatoires

Jean-François Dufourd, Université de Strasbourg.

Les hypercartes combinatoires ont été formellement spécifiées pour prouver interactivement le théorème des 4-couleurs ou le théorème de Jordan discret. Elles sont aussi la base de la certification d'algorithmes fonctionnels en géométrique computationnelle, comme la triangulation de Delaunay. Nous en proposons une spécification constructive et une implantation chaînée en utilisant des orbites de fonctions dans des domaines finis. Tout le développement est formalisé en Coq. Les orbites unifient la présentation aux niveaux conceptuel et concret et réduisent l'effort de preuve. L'implantation est réalisée dans un modèle général de mémoire et les opérations sont écrites en une forme facile à traduire en langage C. Nous prouvons que spécification et implantation sont équivalentes observationnellement. Cette méthode est adaptable à une grande classe de spécifications algébriques implantées dans des structures de données concrètes avec des listes chaînées linéaires, circulaires ou symétriques, enchevêtrées ou emboîtées.

11h30–12h00

A verified information-flow architecture. Delphine Demange, IRISA.

SAFE is a clean-slate effort to build a highly secure computer system that includes pervasive mechanisms for tracking and limiting information flows. At the lowest level, the SAFE hardware supports fine-grained programmable tags, with efficient and flexible propagation and combination of tags as instructions are executed. The operating system virtualizes these generic facilities to present an information-flow abstract machine, on which user programs can label sensitive data with rich confidentiality and integrity policies. We present a formal, machine-checked model of the key information-flow mechanisms of the SAFE hardware and software, together with an end-to-end proof of noninterference for this model. This is joint work with A. Azevedo de Amorim, N. Collins, A. DeHon, C. Hritcu, D. Pichardie, B. C. Pierce, R. Pollack, and A. Tolmach.

12h00–13h00

Buffet

13h00–13h25

Zenon Modulo : quand Achille rattrape la tortue en utilisant la déduction modulo

David Delahaye, CNAM.

Nous proposons une extension de Zenon, un outil de déduction automatique au premier ordre et basé sur les tableaux, à la déduction modulo. La théorie de la déduction modulo est une extension du calcul des prédicats, qui permet de réécrire des termes ainsi que des propositions, et qui est bien adaptée à la recherche de preuve dans les théories axiomatiques, puisqu'elle transforme les axiomes en règles de réécriture. Nous présentons également une heuristique pour effectuer cette transformation automatiquement, et évaluons notre approche au moyen de résultats expérimentaux obtenus à partir des "benchmarks" de la bibliothèque TPTP, où cette heuristique est capable de démontrer des problèmes difficiles dans la théorie des ensembles en particulier. Enfin, nous décrivons un nouveau "backend" pour Zenon capable de produire des certificats de preuve pour Dedukti, qui est un vérificateur de preuves basé sur le lambda-pi-calcul modulo.

13h30–13h55

The Spirit of Ghost Code

Léon Gondelman, LRI.

In the context of deductive program verification, ghost code is part of the program that is added for the purpose of specification. Ghost code must not interfere with regular code, in the sense that it can be erased without any observable difference in the program outcome. In particular, ghost data cannot participate in regular computations and ghost code cannot mutate regular data or diverge. The idea exists in the folklore since the early notion of auxiliary variables and is implemented in many state-of-the-art program verification tools. However, a rigorous definition and treatment of ghost code is surprisingly subtle and few formalizations exist.

We describe a simple ML-style programming language with mutable state and ghost code. Non-interference is ensured by a type system with effects, which allows, notably, the same data types and functions to be used in both regular and ghost code. We define the proce- dure of ghost code erasure and we prove its safety using bisimulation. A similar type system, with numerous extensions which we briefly discuss, is implemented in the program verification environment Why3.

14h00–14h25

Model checking paramétré : invariants et certification

Alain Mebsout, LRI.

Dans cet exposé, on présente un algorithme capable d'inférer automatiquement des invariants assez précis et puissants pour mener à bien la preuve de sûreté de protocoles de taille industrielle comme FLASH. Cet algorithme (BRAB) calcule des sur-approximations de l'ensemble des états atteignables en s'aidant de l'exploration d'instances finies du système. Pour s'assurer de l'absence d'erreurs dans l'implémentation de BRAB, on en fait la vérification formelle dans Why3 puis on extrait le code pour obtenir un model checker efficace et certifié. Ces travaux autour de la certification sont exposés dans une seconde partie.

14h30–14h55

Type Theory in Ludics.

Eugenia Sironi, Marseille.

Ludics is a theory introduced by J.-Y. Girard and comes from a fine analysis of the multiplicative, additive fragment of Linear Logic (MALL).

We present some first steps in the more general setting of the interpretation of dependent type theory in Ludics. Dependent type theories started in the early 1970's, when Martin-Lof introduced his intuitionistic theory of types. The framework is the following : a (Martin-Lof) type A is represented by a behavior (which corresponds to a formula) in such a way that canonical elements of A are interpreted in a set that is principal for the behavior, where principal means in some way a minimal generator. We propose a representation for simple types in Ludics, i.e., natural numbers, lists, the arrow construction for them. Finally we focus on an example of dependent type and propose a generalization for constructors.

15h00–15h30

Pause

15h30–15h55

L'interprétation calculatoire du forcing en logique classique

Lionel Rieg, ENSIIE.

Au travers de la correspondance de Curry-Howard, les traductions CPS (Continuation Passing style) peuvent se voir comme le pendant calculatoire de la non-non traduction de la logique classique dans la logique intuitionniste. Partant de la même idée, on s'intéresse ici à la transformation de programme qui correspond au forcing de Cohen en logique classique.

Dans la lignée des travaux de Jean-Louis Krivine et d'Alexandre Miquel, je vais présenter cette transformation dans le cadre de l'arithmétique d'ordre supérieur ($PA\omega^*$), puis exposer comment elle peut s'intégrer directement à la machine abstraite servant à évaluer les preuves (KAM). Si le temps le permet, j'illustrerai cette méthodologie par l'exemple de l'extraction d'arbres de Herbrand.

16h00–16h25

Coinduction et arbres rationnels.

Régis Spadotti, IRIT

La coinduction est un outil puissant qui permet de raisonner sur des objets infinis. On se propose d'étudier la sous-classe des termes coinductifs ayant un nombre fini de sous-termes. On définira des opérations (foncteurs) qui préservent cette propriété. Enfin, on montrera que certaines propriétés deviennent décidables lorsqu'elles sont restreintes aux éléments de cette sous-classe. Les résultats de ces travaux ont été mécanisés dans l'assistant de preuve Agda.

16h30–16h55

Sections atomiques emboîtées avec échappement de processus légers :une définition formelle

Thomas Pinsard, Orléans *La mémoire transactionnelle est un mécanisme de plus en plus populaire pour la programmation parallèle et concurrente. Dans la plupart des implantations, l'emboîtement de transactions n'est pas possible ce qui pénalise la modularité. Plutôt que les transactions, qui sont un choix possible d'implantation, nous considérons directement la notion de section atomique. Dans un objectif d'améliorer la modularité et l'expressivité, nous considérons un langage impératif simple étendu avec des instructions de parallélisme avec lancement et attente de processus légers et une instruction de section atomique à portée syntaxique, depuis laquelle des processus légers peuvent s'échapper.*

Notre contribution est la définition précise de l'atomicité et de la bonne synchronisation dans ce contexte. Nous prouvons que pour des traces bien formées, la dernière impliquent la forme forte de la première.