Abstract

Esterel is an imperative concurrent design language for the specification of control-oriented reactive systems. Based on the synchronous paradigm, its semantics relies on a clear distinction of instants of computation. All primitive instructions of the language but one "pause" instruction execute in zero time. Execution is thus a sequence of instantaneous computations separated by explicit pauses. Arbitrary loops in this context are troublesome, potentially leading to a non-termination problem or a schizophrenia issue: first, instantaneous loops may prevent the instant to end; second, program blocks may be traversed several times within the same instant, thus having a "schizophrenic" behavior. Instantaneous loops are forbidden by the semantics. Such errors have to be anticipated, and programs rejected by compilers on this behalf. Moreover, efficient code generation for schizophrenic program patterns is complex. While many existing compilers already generate correct code for loops, the efficient implementations available today are neither generic (i.e. target-independent) nor formally specified or verified.

In this work, we thoroughly consider loops in Esterel, starting from the operational semantics of the language, all the way down to a provably correct implementation. We formally characterize the related issues and define efficient static analysis techniques to detect them in Esterel code. In order to get rid of schizophrenic behaviors by source-to-source rewriting – cure schizophrenia – we introduce in Esterel a new primitive instruction, which we call "gotopause". It behaves as a non-instantaneous jump instruction compatible with concurrency. We describe a first program transformation that systematically replaces loops by the mean of gotopause statements, providing a loop-free equivalent program for any correct Esterel program. By combining static analysis and rewriting techniques, we obtain a preprocessor for Esterel that rejects incorrect loops and cure schizophrenia, which we have implemented. Due to our source-to-source transformation methodology, our preprocessor is highly generic; because of static analysis, it is very efficient; thanks to our fully formalized approach, we could formally establish its correctness.

Keywords: synchronous languages, structural operational semantics, concurrency, static analysis, program transformation, correct-by-construction algorithms.

Résumé

Esterel est un langage impératif concurrent pour la programmation des systèmes réactifs. A l'exception de l'instruction "pause", les primitives du langage s'exécutent sans consommer de temps logique. L'exécution se décompose donc en une suite d'instants. Dans ce contexte, les boucles peuvent poser deux types de problèmes: d'une part une boucle instantanée peut bloquer l'écoulement du temps; d'autre part un bloc de code peut être traversé plusieurs fois au cours du même instant, conduisant à un comportement du programme dit "schizophrène". Les boucles instantanées sont proscrites par la sémantique. Elles doivent donc être détectées par les compilateurs et les programmes correspondants doivent être rejetés. Par ailleurs, la compilation efficace des programmes schizophrènes est difficile. Ainsi, alors que plusieurs compilateurs pour Esterel sont disponibles, les algorithmes employés pour compiler les boucles ne sont ni portables, ni formellement spécifiés, et encore moins prouvés.

Dans ce document, nous étudions les boucles en Esterel, établissant une correspondance formelle entre la sémantique opérationnelle du langage et l'implémentation concrète d'un compilateur. Après avoir spécifié les problèmes posés par les boucles, nous développons des techniques d'analyse statique efficaces pour les détecter dans un code Esterel quelconque. Puis, de façon à guérir la schizophrénie, c'est à dire transformer efficacement les programmes schizophrènes en programmes non schizophrènes, nous introduisons dans le langage une nouvelle primitive appelée "gotopause". Elle permet de transférer le contrôle d'un point du programme à un autre de façon non instantanée, mais sans contrainte de localité. Elle préserve le modèle de concurrence synchrone d'Esterel. Nous décrivons un premier algorithme qui, en dépliant les boucles à l'aide de cette nouvelle instruction, produit pour tout programme Esterel correct un programme non schizophrène équivalent. Enfin, en combinant analyse statique et réécriture, nous obtenons un préprocesseur qui rejette les boucles instantanées et guérit la schizophrénie, à la fois portable et très efficace. Nous l'avons implémenté. De plus, grâce à une approche formelle de bout en bout, nous avons pu prouver la correction de ce préprocesseur.

Mots-clés : langages synchrones, sémantique opérationnelle structurelle, concurrence, analyse statique, transformation de programmes, algorithmes corrects par construction.