Necro ML: Kit de Nécromancie

Louis Noizet, Alan Schmitt

Necro ML: Kit de Nécromancie (Démonstration)

Louis Noizet and Alan Schmitt

INRIA Rennes – Bretagne Atlantique

Abstract

On présente Necro ML, un outil pour générer des interpréteurs OCaml à partir de sémantiques en Skel, un langage minimaliste de description de sémantiques. Ces interpréteurs sont modulaires, en exploitant des monades pour interpréter les calculs.

Introduction

Ce travail est initialement basé sur les travaux de Courant *et al.* [3]. En particulier, le langage Skel a été largement amélioré, avec du polymorphisme, des fonctions de première classe, et du filtrage. L'interpréteur OCaml généré est maintenant modulaire pour l'interprétation des branches, ce qui permet d'en changer aisément. La section 1 présente le langage Skel. La section 2 décrit comment un interpréteur est généré. Enfin, la section 3 montre comment les monades d'interprétations peuvent être utilisées pour représenter différents comportements de l'interpréteur. Lors de la démonstration, nous présenterons le fonctionnement de Necro ML et l'usage de différentes monades d'interprétations pour traiter le non-déterminisme.

1 Skel

Skel [2] est un langage statiquement typé pour décrire des sémantiques opérationnelles de langages. Il est à la fois léger et puissant, et une sémantique squelettique peut notamment être extraite vers un générateur OCaml, une formalisation Coq, et un débogueur. Le langage Skel est basé sur ML avec une construction supplémentaire appelée le branchement pour traiter les calculs non-déterministes, et la possibilité de sous-spécifier.

Une des fonctionnalités principales de Skel est la séparation entre *squelettes* et *termes*. Les premiers représentent des calculs, et les seconds des valeurs. Cela est inspiré du λ-calcul computationnel de Moggi [6].

Skel est proche de langages ML comme OCaml ou SML, mais son AST est léger, de manière à pouvoir être manipulé plus simplement. Il est défini dans [7], et contient uniquement 126 lignes de spécification, ce qui permet de gérer et extraire facilement des sémantiques.

2 Necro ML

Necro est un écosystème pour manipuler les sémantiques squelettiques. Il repose sur Necro Lib [7], une bibliothèque OCaml qui définit la syntaxe de Skel, et un ensemble de fonctions pour manipuler les sémantiques squelettiques, y compris la fonction `parse_and_type`, qui lit un fichier et renvoie une sémantique squelettique typée.

Ainsi, n'importe qui a accès à la syntaxe et à des fonctions de base, pour pouvoir créer des outils qui exploitent les sémantiques squelettiques et étendent l'écosystème Necro. En plus de Necro Lib, nous fournissons plusieurs outils. L'un d'eux, celui que nous présentons ici, est Necro ML [5], qui génère des interpréteurs OCaml depuis des sémantiques squelettiques.

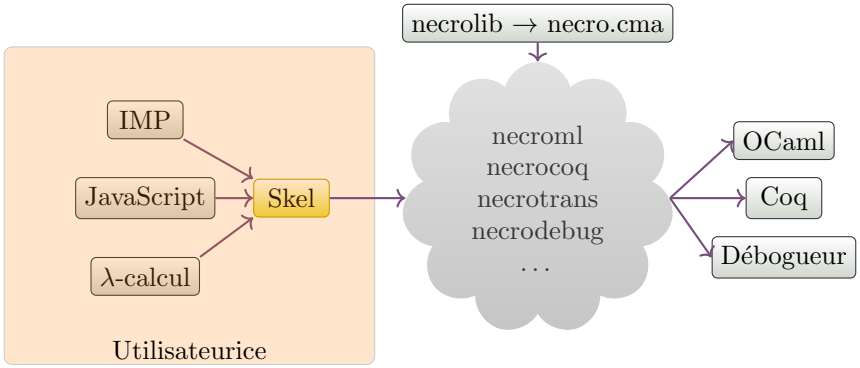

Figure 1: The Necro ecosystem

Comme nous l'avons précisé dans la section 1, il y a dans les sémantiques squelettiques des données non-spécifiées. Cela est traité par Necro ML de manière modulaire. On génère un foncteur `MakeInterpreter`, qui produit un interpréteur quand on lui fournit une implémentation des termes et types non-spécifiés.

Le plongement est principalement superficiel, mais on doit gérer les branches avec prudence, puisque OCaml n'a pas de construction native équivalente. Nous allons donc montrer comment cela est traité dans la section 3.

3 Monade d'interprétation

Comme expliqué plus haut, on ne peut pas utiliser un simple plongement profond, en raison des constructions non-déterministes. À cette fin, on utilise une monade de plongement, ou monade d'interprétation, pour décrire les calculs. Ainsi, les termes de type `'a` sont plongés superficiellement dans des expressions OCaml de type `'a`, tandis que les squelettes de type `'a` sont plongés profondément dans des expressions OCaml de type `'a M.t`. Necro ML propose plusieurs monades d'interprétations, et d'autres peuvent être définies par l'utilisateurice. L'aspect principal est qu'il est très simple de changer la monade d'interprétation sans avoir à réécrire aucun code.

4 Conclusion

Necro ML est un outil flexible pour générer des interpréteurs. Il effectue un plongement superficiel des types, et semi-profond des expressions pour gérer le non-déterminisme. Il a été testé sur des fichiers de taille significative, notamment dans le travail en cours de formalisation de JavaScript [4].

D'autre projets sont intimement liés. D'abord, Necro Debug est une exécution pas-à-pas d'une sémantique squelettique.[1] Il utilise la même approche et les même signatures de modules que Necro ML, ce qui permet de réutiliser l'instanciation des termes non-spécifiés. Ensuite, Ambal *et al.* ont présenté un générateur d'interpréteurs certifiés à l'aide de Necro Coq et du mécanisme d'extraction [1].

[1] `https://skeletons.inria.fr/debugger/index_while.html`

References

[1] Guillaume Ambal, Sergueï Lenglet, and Alan Schmitt. Certified Abstract Machines for Skeletal Semantics. In *Certified Programs and Proofs*, Philadelphia, United States, January 2022.

[2] Martin Bodin, Philippa Gardner, Thomas Jensen, and Alan Schmitt. Skeletal Semantics and their Interpretations. *Proceedings of the ACM on Programming Languages*, 44:1–31, 2019. URL: https://hal.inria.fr/hal-01881863, doi:10.1145/3290357.

[3] Nathanaëlle Courant, Enzo Crance, and Alan Schmitt. Necro: Animating Skeletons. In *ML 2019*, Berlin, Germany, Aug 2019.

[4] Adam Khayam, Louis Noizet, and Alan Schmitt. JSkel: Towards a Formalization of JavaScript's Semantics, 2022. Submitted for publication.

[5] Enzo Crance Martin Bodin, Nathanaëlle Courant and Louis Noizet. Necro Ocaml Generator. URL: https://gitlab.inria.fr/skeletons/necro-ml.

[6] Eugenio Moggi. Computational lambda-calculus and monads. *Proceedings. Fourth Annual Symposium on Logic in Computer Science*, 1988.

[7] Louis Noizet. Necro Library. URL: https://gitlab.inria.fr/skeletons/necro.