Catala, un langage pour transformer la loi en code (démonstration)

Alain Delaët, Denis Merigoux

▶ **To cite this version:**

Alain Delaët, Denis Merigoux. Catala, un langage pour transformer la loi en code (démonstration). 33èmes Journées Francophones des Langages Applicatifs, Jun 2022, Saint-Médard-d'Excideuil, France. pp.264-266. hal-03626853

HAL Id: hal-03626853
https://inria.hal.science/hal-03626853v1

Submitted on 31 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Catala, un langage pour transformer la loi en code

Alain Delaët[1,2] and Denis Merigoux[1]

[1] Inria Paris
[2] ENS de Lyon

Résumé

Le droit codifie et régit de nombreux aspects de la vie quotidienne. Si la plupart du temps les lois sont sujettes à interprétation, dans certains domaines la loi ne permet pas d'interprétation et vise essentiellement à décrire rigoureusement un calcul ou une procédure de décision, c'est-à-dire un algorithme.

Catala est un nouveau langage conçu en collaboration avec des juristes qui permet une compilation depuis une spécification proche de la loi vers un code exécutable. Contrairement aux langages traditionnels, Catala est adapté aux raisonnements cas de base/exceptions présents dans la loi ; il intègre de la programmation littéraire du droit qui facilite les mises à jour du programme. Finalement, Catala compile vers une variété de langages cibles dont OCaml.

Dans cette démonstration de prototype, nous montrerons comment exprimer une partie du calcul des allocations familiales dans Catala.

Contexte. Dans certains domaines juridiques, les textes décrivent comment le calcul d'une quantité doit être effectué. C'est par exemple le cas pour le droit fiscal et le droit des prestations familiales. L'administration doit cependant se charger de l'implémentation effective du processus, c'est-à-dire de transformer la loi en code. La méthode actuelle se décompose en trois étapes [4]. La première correspond à l'écriture de la loi, sous forme exclusivement littéraire, qui est assurée par le législateur. Ensuite, le service juridique de la direction générale des finances publiques doit interpréter le texte, en effectuant des recherches juridiques au besoin, pour en extraire une spécification. Finalement, la construction du logiciel mettant en œuvre cette spécification est soit assurée par le service informatique de l'administration, soit sous-traitée à des entreprises externes, et le logiciel résultant est utilisé en production pour le calcul des impôts par exemple.

Cette démarche pose plusieurs problèmes. Premièrement, la correction fonctionnelle – le fait que le calcul effectué correspond bien à sa spécification juridique – est vérifiée uniquement sur quelques exemples de tests [1]. Ces "cas pratiques" sont rédigés par des juristes qui n'ont pas forcément le temps ni la méthodologie pour tester tous les cas possibles, si bien que certains cas peuvent arriver en production sans avoir été totalement testés. Deuxièmement, lors d'un changement législatif, l'ensemble du processus doit être repris, car une petite modification dans la législation peut changer en profondeur la façon dont le calcul peut être réalisé dans un langage traditionnel. Par exemple, le gouvernement a invoqué des difficultés d'implémentation qui empêcheraient de déconjugaliser l'allocation adultes handicapés [5].

Ces besoins de fiabilité et d'incrémentalité sont des problèmes étudiés en informatique, et les méthodes formelles y apportent partiellement des solutions. Ainsi, la vérification de programme a déjà fait ses preuves pour les programmes critiques.

Introduction d'un nouveau DSL Nous proposons une nouvelle méthode de travail qui s'appuie sur un nouveau langage de programmation dédié, Catala, introduit dans [3]. Le langage a été conçu en collaboration avec des juristes dans le but d'exprimer des spécifications dans un style ressemblant à celui des articles de loi qui servent de référence au programme.

Les fonctionnalités de Catala visent à permettre une relecture du code de la part de professionnels du droit. Par exemple, la syntaxe de Catala est très verbeuse, ce qui facile sa lecture par les juristes. Son *parser* dispose de plusieurs *lexers* pour faciliter le support d'une nouvelle langue : français, anglais et polonais sont actuellement disponibles. Finalement, un programme Catala est composé de code et de prose, utilisant de la programmation littéraire. Les juristes peuvent directement réfléchir sur le code Catala plutôt que sur des cas pratiques, de la même façon qu'un programmeur raisonne sur la sémantique de son programme plutôt que sur des tests.

Pour écrire de nouveaux programmes en Catala, l'approche que nous avons choisie a été de faire de la programmation par binômes pluridisciplinaires. Le travail de l'informaticienne est de repérer les ambiguïtés et de demander au juriste de les résoudre par un raisonnement juridique. Le juriste peut, grâce à la lisibilité du code, vérifier que le code écrit correspond bien à l'esprit et à la lettre de la loi.

La programmation littéraire, en plus de pouvoir générer des documents compatibles avec les habitudes des juristes, relie article par article la loi avec le code Catala. Cela permet d'effectuer localement les modifications rendues nécessaires par un amendement de la loi. Cette approche est garante de l'incrémentalité mentionnée ci-dessus.

Finalement, l'ensemble du compilateur est Open Source et implémenté en OCaml. Conformément aux dispositions législatives sur la publication des codes sources comme documents administratifs, il est également nécessaire que les programmes Catala utilisés par d'éventuelles administrations soient également Open Source.

Caractéristiques du langage Formellement, Catala s'appuie sur la logique par défaut, formalisée en 1980 par [6]. Cette logique sous-tend la structure de la loi, comme montrée en 2018 par [2]. La représentation intermédiaire centrale de Catala est un λ-calcul augmenté d'un terme modélisant une structure cas de base/exceptions, conformément à la logique par défaut.

L'autre spécificité de Catala est sa chaine de compilation qui permet de compiler le code Catala vers plusieurs langages cibles : Python, OCaml et JavaScript (via js_of_ocaml). Ce schéma de compilation permet une interopérabilité virtuellement illimitée, avec par exemple des langages ne disposant même pas de FFI, ou des architectures pré-API.

Pour une description complète du langage, se reporter à [3].

Contenu de la démonstration Dans cette démonstration, nous monterons comment exprimer une partie du calcul des allocations familiales dans Catala. Nous exposerons notamment le travail de recherche juridique qui a été nécessaire pour lever les ambiguïtés des formulations. Nous verrons ensuite comment Catala génère du code exécutable, en suivant sa transformation jusqu'à la mise en production.

Références

[1] Liane Huttner and Denis Merigoux. Traduire la loi en code grâce au langage de programmation Catala. *Revue de droit fiscal*, (5) :121, 2021.

[2] Sarah B. Lawsky. A Logic for Statutes. *Florida Tax Review*, 2018.

[3] Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. Catala : A programming language for the law. *Proc. ACM Program. Lang.*, 5(ICFP), August 2021.

[4] Denis Merigoux, Raphaël Monat, and Jonathan Protzenko. A modern compiler for the french tax code. In *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*, CC 2021, page 71–82, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Assemblée nationale. Débats sur la proposition de loi portant diverses mesures de justice sociale. https://www.assemblee-nationale.fr/dyn/15/comptes-rendus/seance/session-ordinaire-de-2020-2021/premiere-seance-du-jeudi-17-juin-2021#2557908, 2021.

[6] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1) :81 – 132, 1980. Special Issue on Non-Monotonic Logic.