Les langages dédiés comme cheval de troie des méthodes formelles

Denis Merigoux

SIESTE – 6 novembre 2024

Inria Paris

Introduction

Le logiciel critique

Plus un logiciel est important, plus un bug est catastrophique !

Le logiciel critique

Plus un logiciel est important, plus un bug est catastrophique !

Le logiciel critique

Plus un logiciel est important, plus un bug est catastrophique !

Le logiciel critique

Plus un logiciel est important, plus un bug est catastrophique !

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ?

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ? Oui !

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ? Oui !

Réussites industrielles

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ? Oui !

Réussites industrielles

ASTRÉE
Analyse statique
de code C critique
temps réel synchrone
embarqué

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ? Oui !

Réussites industrielles

ASTRÉE
Analyse statique
de code C critique
temps réel synchrone
embarqué

Les méthodes formelles

Peut-on prouver l'absence de certains bugs ? Oui !

Réussites industrielles

ASTRÉE
Analyse statique
de code C critique
temps réel synchrone
embarqué

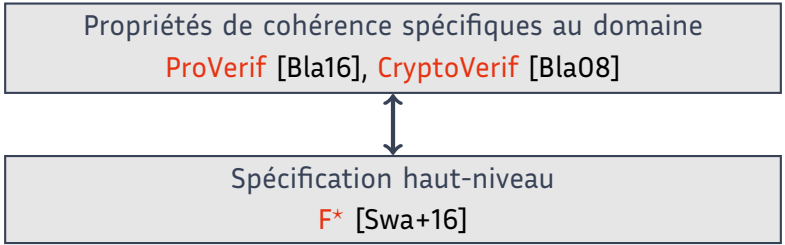
CompCert [Ler06]

Un exemple d'architecture de vérification de programmes

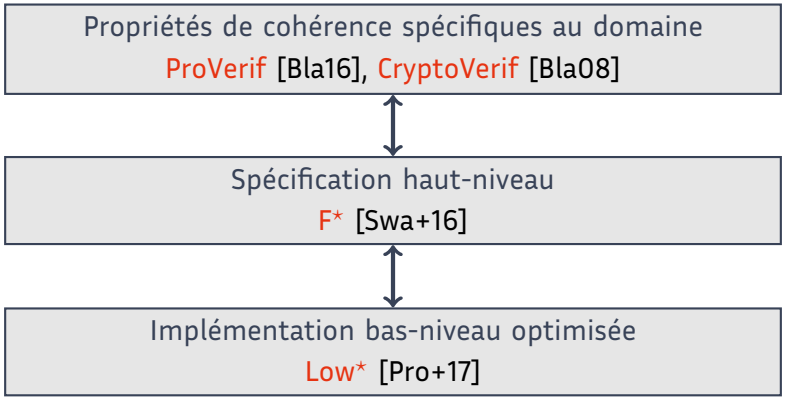
Spécification haut-niveau

F* [Swa+16]

Un exemple d'architecture de vérification de programmes



Un exemple d'architecture de vérification de programmes



Un assistant de preuve pour les gouverner tous ?

Un assistant de preuve pour les gouverner tous ?

Un outillage complet et intégré pour la vérification de programmes en Rocq

Un assistant de preuve pour les gouverner tous ?

Un outillage complet et intégré pour la vérification de programmes en Rocq

- Tous les langages sont définis et prouvés méta-théoriquement en Rocq

Un assistant de preuve pour les gouverner tous ?

Un outillage complet et intégré pour la vérification de programmes en Rocq

- Tous les langages sont définis et prouvés méta-théoriquement en Rocq
- La compilation est implémentée et prouvée en Rocq

Un assistant de preuve pour les gouverner tous ?

Un outillage complet et intégré pour la vérification de programmes en Rocq

- Tous les langages sont définis et prouvés méta-théoriquement en Rocq
- La compilation est implémentée et prouvée en Rocq
- Les programmes sont écrits en Rocq dans le *deep embedding* du langage

Un assistant de preuve pour les gouverner tous ?

Un outillage complet et intégré pour la vérification de programmes en Rocq

- Tous les langages sont définis et prouvés méta-théoriquement en Rocq
- La compilation est implémentée et prouvée en Rocq
- Les programmes sont écrits en Rocq dans le *deep embedding* du langage

Méthodologie illustrée par : Fiat [Del+15 ; Chl+17]

Un assistant de preuve pour les gouverner tous ?

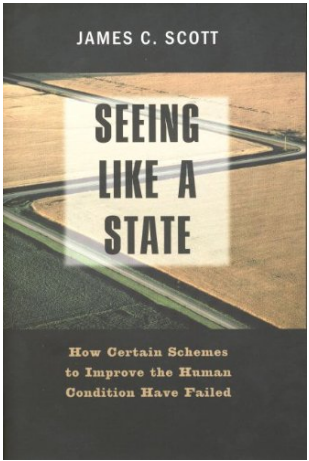
Un outillage complet et intégré pour la vérification de programmes en Rocq

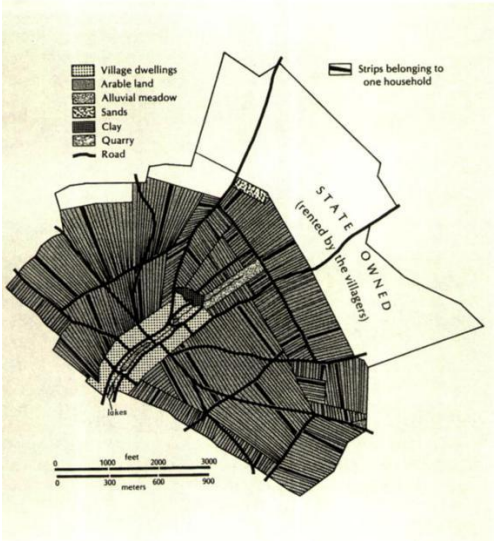
- Tous les langages sont définis et prouvés méta-théoriquement en Rocq
- La compilation est implémentée et prouvée en Rocq
- Les programmes sont écrits en Rocq dans le *deep embedding* du langage

Méthodologie illustrée par : Fiat [Del+15 ; Chl+17]

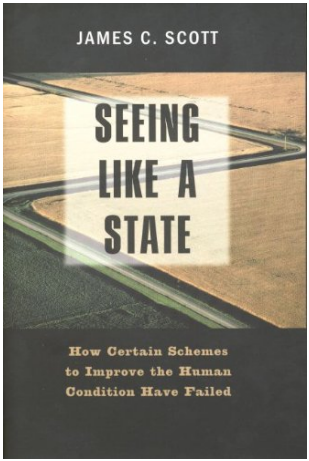
```
Definition BookStoreSpec : ADT BookStoreSig :=
  QueryADTRep BookStoreSchema {
    query "NumOrders" ( author : string ) : nat :=
       Count (For (o in sORDERS) (b in sBOOKS)
              Where (author = b!sAUTHOR)
              Where (b!sISBN = o!sISBN)
              Return ())  }.
```

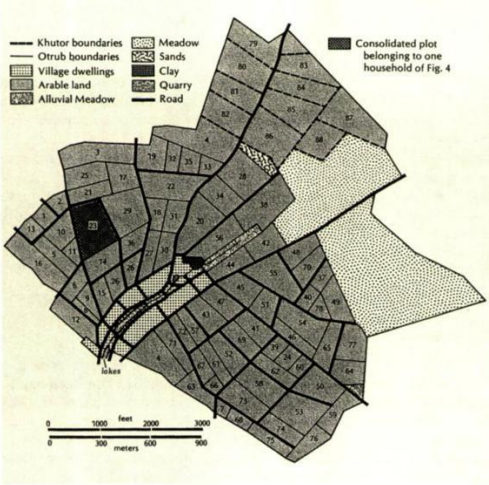
Petit détour par les réformes agraires du régime tsariste





Petit détour par les réformes agraires du régime tsariste

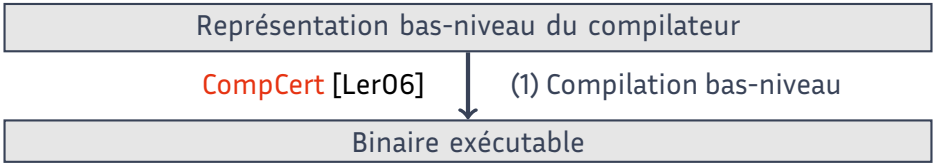




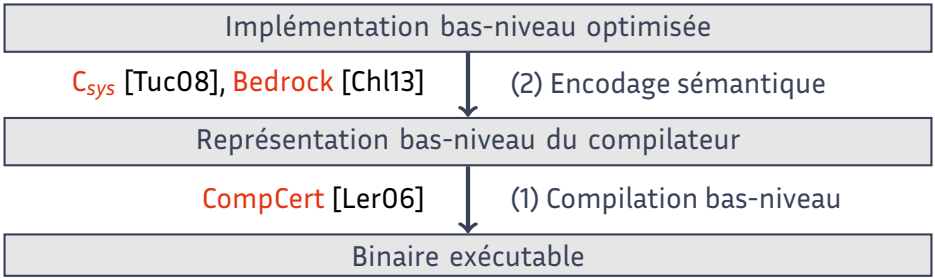
La chaîne de confiance : où est le maillon faible ?

Binaire exécutable

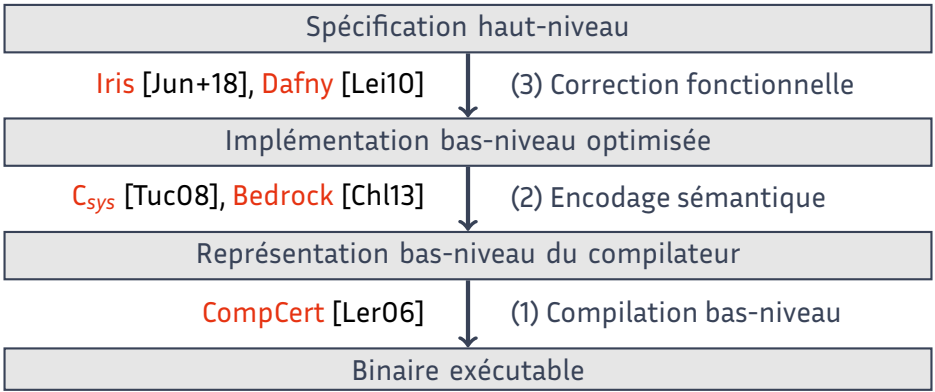
La chaîne de confiance : où est le maillon faible ?



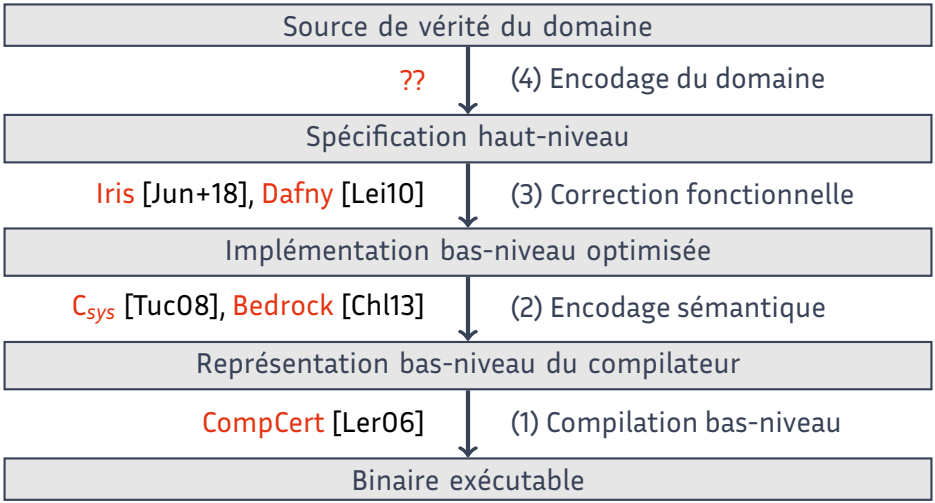
La chaîne de confiance : où est le maillon faible ?



La chaîne de confiance : où est le maillon faible ?



La chaîne de confiance : où est le maillon faible ?



Connecter preuves et programmes à l'aide de langages agiles

Langages dédiés orientés vers la preuve

Pour

Contre

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis

Contre

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat

Contre

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat

Contre

– Réinventer la roue

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat

Contre

– Réinventer la roue
– Manque d'accès aux bibliothèques populaires

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat
+ Implémentation externe/interne

Contre

– Réinventer la roue
– Manque d'accès aux bibliothèques populaires

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat
+ Implémentation externe/interne

Contre

– Réinventer la roue
– Manque d'accès aux bibliothèques populaires
– Besoin d'expertise spécifique

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat
+ Implémentation externe/interne
+ Ateliers de création de langages

Contre

– Réinventer la roue
– Manque d'accès aux bibliothèques populaires
– Besoin d'expertise spécifique

Langages dédiés orientés vers la preuve

Pour

+ Programmes concis
+ Paradigme logique adéquat
+ Implémentation externe/interne
+ Ateliers de création de langages

Contre

– Réinventer la roue
– Manque d'accès aux bibliothèques populaires
– Besoin d'expertise spécifique
– Petites communautés

Langages dédiés orientés vers la preuve

Pour

- \+ Programmes concis
- \+ Paradigme logique adéquat
- \+ Implémentation externe/interne
- \+ Ateliers de création de langages

Contre

- – Réinventer la roue
- – Manque d'accès aux bibliothèques populaires
- – Besoin d'expertise spécifique
- – Petites communautés

Théo Zimmermann [Zim19] :

> *« […] Rocq ne bénéficie pas des moyens massifs auxquels les concepteurs de langages employés par de grosses entreprises sont habitués. […] un haut niveau d'expertise est requis pour contribuer […] »*

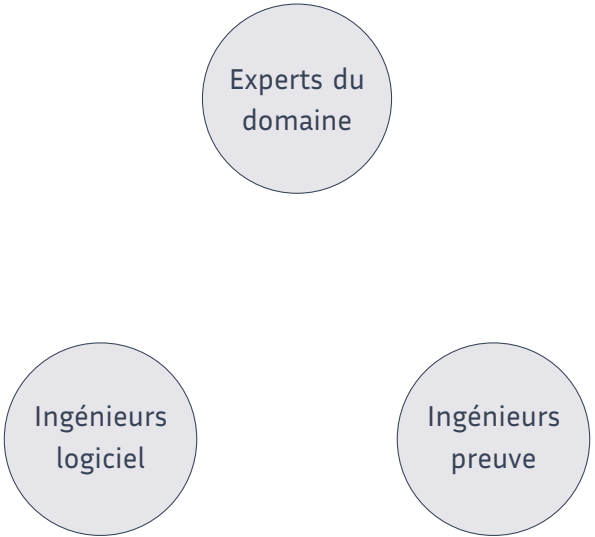
Une méthodologie pour repousser la frontière de la vérification

Une méthodologie pour repousser la frontière de la vérification

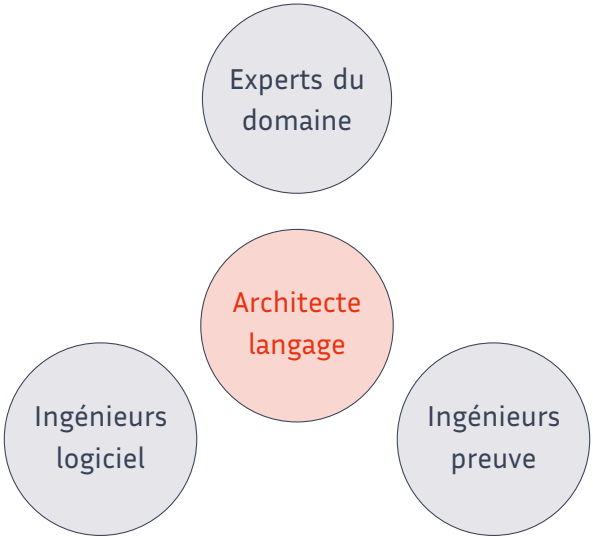
Objectif

Nous voulons appliquer les méthodes formelles à des bases de code existantes de logiciel critique pour augmenter leur niveau d'assurance.

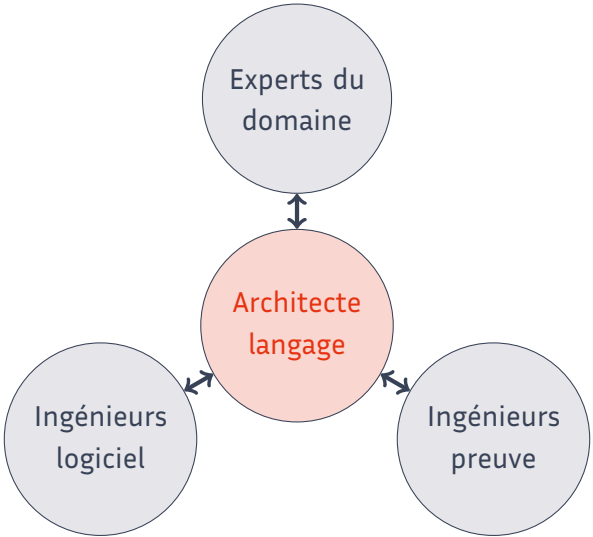
Le manifeste de l'architecte langage



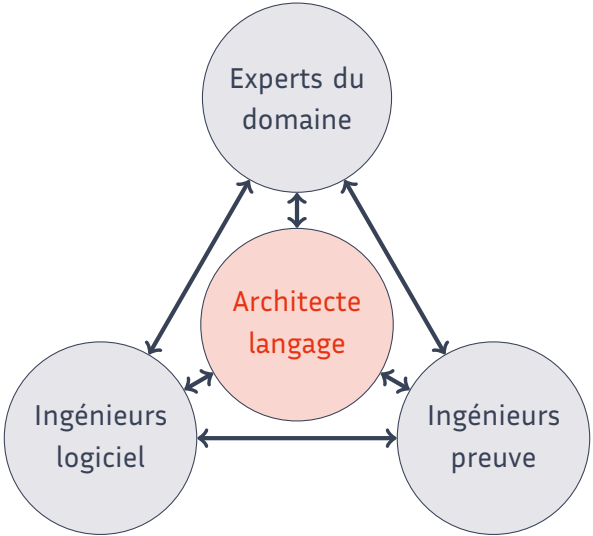
Le manifeste de l'architecte langage



Le manifeste de l'architecte langage



Le manifeste de l'architecte langage



Résumé de mes contributions de thèse

Résumé de mes contributions de thèse

Cryptographie

Résumé de mes contributions de thèse

Cryptographie

LibSignal* Amener la cryptographie personnalisée sur le Web [Pro+19]

Résumé de mes contributions de thèse

Cryptographie

LibSignal*	Amener la cryptographie personnalisée sur le Web [Pro+19]
hacspec	Rendre les spécifications cryptographiques formelles accessibles [MKB21]

Résumé de mes contributions de thèse

Cryptographie

LibSignal*	Amener la cryptographie personnalisée sur le Web [Pro+19]
hacspec	Rendre les spécifications cryptographiques formelles accessibles [MKB21]
Steel	Gérer les obligations de preuve pour les programmes bas-niveau [Swa+20; Fro+21]

Résumé de mes contributions de thèse

Cryptographie

LibSignal* Amener la cryptographie personnalisée sur le Web [Pro+19]

hacspec Rendre les spécifications cryptographiques formelles accessibles [MKB21]

Steel Gérer les obligations de preuve pour les programmes bas-niveau [Swa+20; Fro+21]

Systèmes d'information légaux

Résumé de mes contributions de thèse

Cryptographie

LibSignal*	Amener la cryptographie personnalisée sur le Web [Pro+19]
hacspec	Rendre les spécifications cryptographiques formelles accessibles [MKB21]
Steel	Gérer les obligations de preuve pour les programmes bas-niveau [Swa+20; Fro+21]

Systèmes d'information légaux

Mlang	Améliorer la fiabilité du calcul de l'impôt sur le revenu [MMP21]

Résumé de mes contributions de thèse

Cryptographie

LibSignal*	Amener la cryptographie personnalisée sur le Web [Pro+19]
hacspec	Rendre les spécifications cryptographiques formelles accessibles [MKB21]
Steel	Gérer les obligations de preuve pour les programmes bas-niveau [Swa+20; Fro+21]

Systèmes d'information légaux

Mlang	Améliorer la fiabilité du calcul de l'impôt sur le revenu [MMP21]
Catala	Permettre un travail commun de formalisation avec des juristes [MCP21]

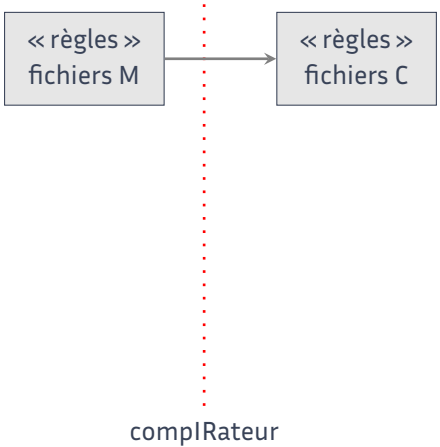
Dans les tranchées du calcul de l'impôt

Architecture technique du calcul de l'impôt sur le revenu en France

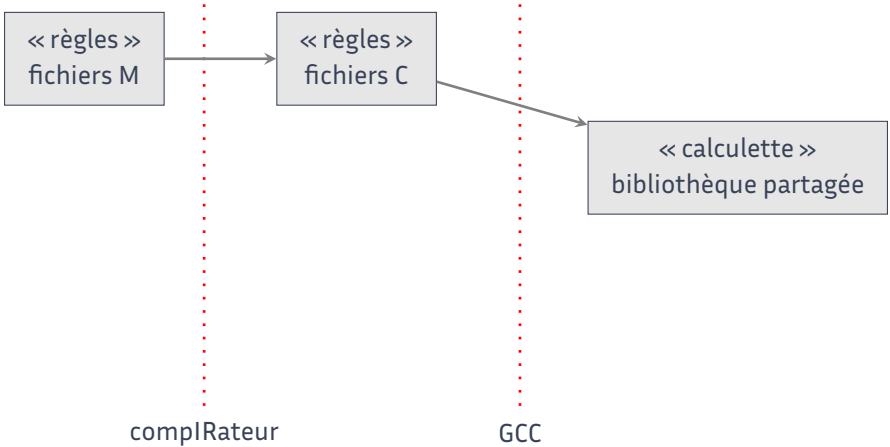
« règles »
fichiers M

Architecture technique du calcul de l'impôt sur le revenu en France

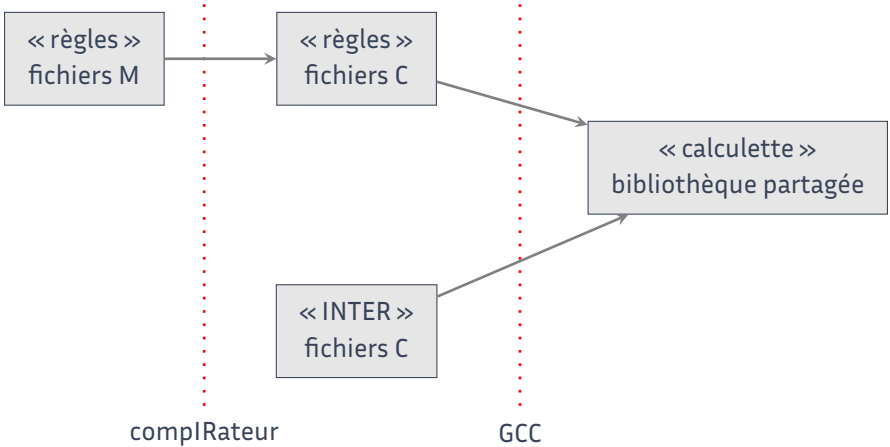
Architecture technique du calcul de l'impôt sur le revenu en France



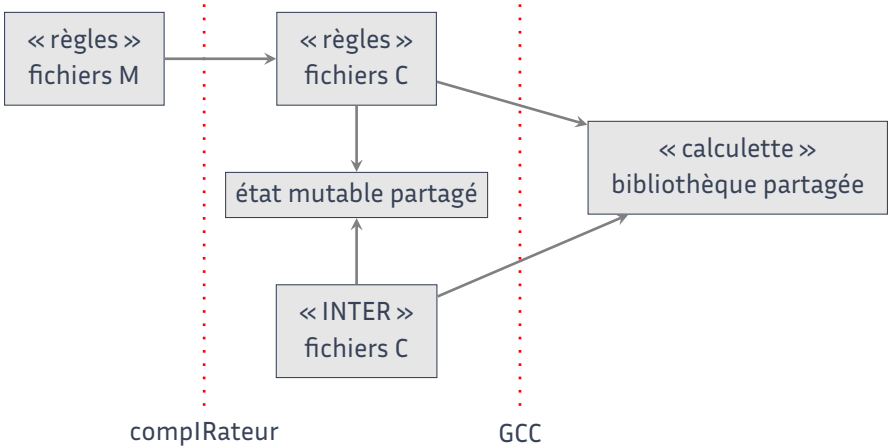
Architecture technique du calcul de l'impôt sur le revenu en France



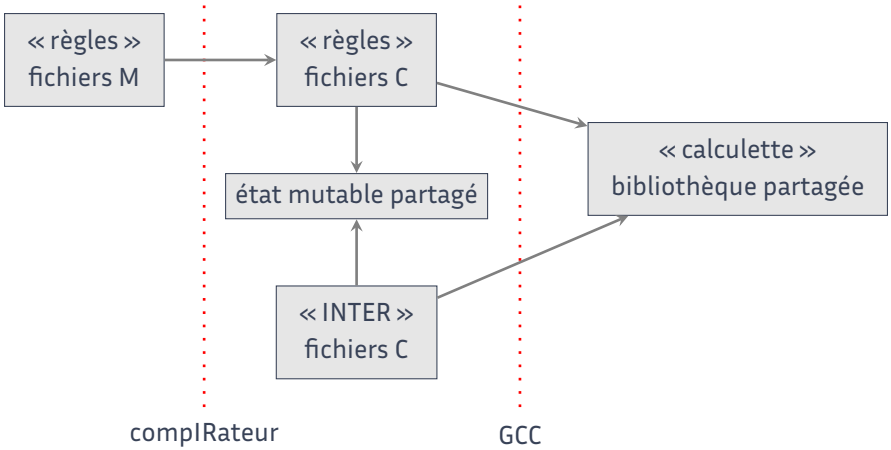
Architecture technique du calcul de l'impôt sur le revenu en France



Architecture technique du calcul de l'impôt sur le revenu en France



Architecture technique du calcul de l'impôt sur le revenu en France



Nouveau compilateur pour M étendu, rétro-ingénierie des fichiers INTER : [MMP21]

Et si on faisait ça proprement plutôt ?

Quelle mode de production de ces logiciels ?

Cycle en V ✗

Quelle mode de production de ces logiciels ?

Cycle en V ✗

Méthodes agiles ✓

Réunir le droit et le code

Idée de base

Il faut entremêler le droit et le code qui le traduit pour que juristes et programmeurs travaillent sur le même document [**bench1992isomorphism**].

Réunir le droit et le code

Idée de base

Il faut entremêler le droit et le code qui le traduit pour que juristes et programmeurs travaillent sur le même document [**bench1992isomorphism**].

$\Rightarrow$ **quel langage de programmation ?**

La structure logique de la loi

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour les contribuables dont les revenus du foyer fiscal, au sens du 1° du IV de l'article 1417, excèdent 19 176 €, pour la première part de quotient familial des personnes célibataires, veuves ou divorcées, ou 38 352 €, pour les deux premières parts de quotient familial des personnes soumises à une imposition commune, ces seuils étant majorés le cas échéant dans les conditions prévues au même premier alinéa, le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre : […]

CGI, article 197, I, 4, b, 3° (circa 2019)

La structure logique de la loi

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour les contribuables dont les revenus du foyer fiscal, au sens du 1° du IV de l'article 1417, excèdent 19 176 €, pour la première part de quotient familial des personnes célibataires, veuves ou divorcées, ou 38 352 €, pour les deux premières parts de quotient familial des personnes soumises à une imposition commune, ces seuils étant majorés le cas échéant dans les conditions prévues au même premier alinéa, le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre : […]

CGI, article 197, I, 4, b, 3° (circa 2019)

[Cas de base : condition]

La structure logique de la loi

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour les contribuables dont les revenus du foyer fiscal, au sens du 1° du IV de l'article 1417, excèdent 19 176 €, pour la première part de quotient familial des personnes célibataires, veuves ou divorcées, ou 38 352 €, pour les deux premières parts de quotient familial des personnes soumises à une imposition commune, ces seuils étant majorés le cas échéant dans les conditions prévues au même premier alinéa, le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre : […]

CGI, article 197, I, 4, b, 3° (circa 2019)

[Cas de base : condition] ⇒ [Cas de base : conséquence]

La structure logique de la loi

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour les contribuables dont les revenus du foyer fiscal, au sens du 1° du IV de l'article 1417, excèdent 19 176 €, pour la première part de quotient familial des personnes célibataires, veuves ou divorcées, ou 38 352 €, pour les deux premières parts de quotient familial des personnes soumises à une imposition commune, ces seuils étant majorés le cas échéant dans les conditions prévues au même premier alinéa, le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre : […]

CGI, article 197, I, 4, b, 3° (circa 2019)

[Cas de base : condition] ⇒ [Cas de base : conséquence]
[Exception : condition]

La structure logique de la loi

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour les contribuables dont les revenus du foyer fiscal, au sens du 1° du IV de l'article 1417, excèdent 19 176 €, pour la première part de quotient familial des personnes célibataires, veuves ou divorcées, ou 38 352 €, pour les deux premières parts de quotient familial des personnes soumises à une imposition commune, ces seuils étant majorés le cas échéant dans les conditions prévues au même premier alinéa, le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre : […]

CGI, article 197, I, 4, b, 3° (circa 2019)

[Cas de base : condition] ⇒ [Cas de base : conséquence]

[Exception : condition] ⇒ [Exception : conséquence]

La logique par défaut à priorités

logiques non-monotone $\supset$ logique défaisable $\supset$ logique par défaut

La logique par défaut à priorités

logiques non-monotone $\supset$ logique défaisable $\supset$ logique par défaut

$$\frac{a \ : \ b_1, \ldots, b_n}{c}$$

Si a est vrai, et si indépendamment, $b_1, \ldots, b_n$ sont cohérents avec a, alors c est vrai.

La logique par défaut à priorités

logiques non-monotone $\supset$ logique défaisable $\supset$ logique par défaut

$$\frac{a \ : \ b_1, \ldots, b_n}{c}$$

Si a est vrai, et si indépendamment, $b_1, \ldots, b_n$ sont cohérents avec a, alors c est vrai.

[Law18] : le droit (fiscal) suit une logique par défaut avec un ordre partiel de priorité sur les défauts à appliquer.

La logique par défaut à priorités

logiques non-monotone $\supset$ logique défaisable $\supset$ logique par défaut

$$\frac{a \ : \ b_1, \ldots, b_n}{c}$$

Si a est vrai, et si indépendamment, $b_1, \ldots, b_n$ sont cohérents avec a, alors c est vrai.

[Law18] : le droit (fiscal) suit une logique par défaut avec un ordre partiel de priorité sur les défauts à appliquer.

$\Rightarrow$ **comment intégrer cela dans un langage de programmation ?**

Le calcul par défaut

Type	τ	$::=$	`bool` $\mid$ `unit`	types unités et booléen
		$\mid$	$\tau \to \tau$	type fonction
Expression	e	$::=$	$x \mid$ `true` $\mid$ `false` $\mid$ `()`	variable, litéral
		$\mid$	$\lambda\,(x : \tau)\,.\,e \mid e\ e$	λ-calcul

Le calcul par défaut

Type	τ	$::=$	$\texttt{bool} \mid \texttt{unit}$	types unités et booléen
		$\mid$	$\tau \to \tau$	type fonction
Expression	e	$::=$	$x \mid \texttt{true} \mid \texttt{false} \mid ()$	variable, litéral
		$\mid$	$\lambda\,(x : \tau)\,.\,e \mid e\ e$	λ-calcul
		$\mid$	d	terme par défaut
Default	d	$::=$	$\langle e^* \mid e :\!- e \rangle$	terme par défaut
		$\mid$	$\circledast$	erreur de conflit
		$\mid$	$\varnothing$	erreur de vide

Le calcul par défaut

Type	τ	$::=$	$\texttt{bool} \mid \texttt{unit}$	types unités et booléen
		$\mid$	$\tau \to \tau$	type fonction
Expression	e	$::=$	$x \mid \texttt{true} \mid \texttt{false} \mid ()$	variable, litéral
		$\mid$	$\lambda\,(x : \tau)\,.\,e \mid e\ e$	λ-calcul
		$\mid$	d	terme par défaut
Default	d	$::=$	$\langle e^* \mid e :\!- e \rangle$	terme par défaut
		$\mid$	$\circledast$	erreur de conflit
		$\mid$	$\varnothing$	erreur de vide

Pour plus de détails, voir [MCP21] !

Sémantique du terme par défaut

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} :\!- e \rangle \longrightarrow \varnothing$$

Sémantique du terme par défaut

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} :- e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} :- v \rangle \longrightarrow v$$

Sémantique du terme par défaut

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} :\!- e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} :\!- v \rangle \longrightarrow v$$

$$\langle \varnothing, \ldots, \varnothing, v, \varnothing, \ldots, \varnothing \mid e_1 :\!- e_2 \rangle \longrightarrow v$$

Sémantique du terme par défaut

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} :- e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} :- v \rangle \longrightarrow v$$

$$\langle \varnothing, \ldots, \varnothing, v, \varnothing, \ldots, \varnothing \mid e_1 :- e_2 \rangle \longrightarrow v$$

$$\frac{\text{nombre\_non\_vide}(v_1, \ldots, v_n) > 1}{\langle v_1, \ldots, v_n \mid e_1 :- e_2 \rangle \longrightarrow \circledast}$$

Enrobage du langage d'expressions

État de l'art

Point commun des langages M, OpenFisca, Oracle Intelligent Advisor, etc.

Enrobage du langage d'expressions

État de l'art

Point commun des langages M, OpenFisca, Oracle Intelligent Advisor, etc.

$\Rightarrow$ équations $x = e$ avec réordonnancement par dépendance, données non-structurées

Enrobage du langage d'expressions

État de l'art

Point commun des langages M, OpenFisca, Oracle Intelligent Advisor, etc.

$\Rightarrow$ équations $x = e$ avec réordonnancement par dépendance, données non-structurées

Où sont les abstractions ? Problème des « doubles liquidation ».

Enrobage du langage d'expressions

État de l'art

Point commun des langages M, OpenFisca, Oracle Intelligent Advisor, etc.

$\Rightarrow$ équations $x = e$ avec réordonnancement par dépendance, données non-structurées

Où sont les abstractions ? Problème des « doubles liquidation ».

Le champ d'application

Champ d'application $\sim$ fonction :

- variables locales déclarées à l'avance et réordonnancées par dépendance
- appels à des sous-champ d'application
- signature fortement typée
- correspondance avec un concept juridique

Un exemple de Catala

Article L521-3 du code de la sécurité sociale

Chacun des enfants à charge, à l'exception du plus âgé, ouvre droit à partir d'un âge minimum à une majoration des allocations familiales.

Article L521-3 du code de la sécurité sociale

Chacun des enfants à charge, à l'exception du plus âgé, ouvre droit à partir d'un âge minimum à une majoration des allocations familiales.

```catala
champ d'application AllocationsFamiliales :
  règle droit_ouvert_majoration de enfant sous condition
    (non (est_enfant_le_plus_âgé de enfant)) et
    (enfant.âge   âge_limite_alinéa_1_l521_3 de enfant) conséquence rempli
```

Article L521-3 du code de la sécurité sociale

Toutefois, les personnes ayant un nombre déterminé d'enfants à charge bénéficient de ladite majoration pour chaque enfant à charge à partir de l'âge mentionné au premier alinéa.

Un exemple de Catala

Article L521-3 du code de la sécurité sociale

Toutefois, les personnes ayant un nombre déterminé d'enfants à charge bénéficient de ladite majoration pour chaque enfant à charge à partir de l'âge mentionné au premier alinéa.

```catala
champ d'application AllocationsFamiliales :
  exception règle droit_ouvert_majoration de enfant sous condition
    (nombre de enfants_à_charge_droit_ouvert_prestation_familiale >=
      nombre_enfants_alinéa_2_l521_3) et
    (enfant.âge >= âge_limite_alinéa_1_l521_3 de enfant) conséquence rempli
```

Co-conception avec les juristes

Beaucoup de juristes dans la littérature *AI and Law*, venant de l'école formaliste en droit qui s'oppose à l'école réaliste.

Co-conception avec les juristes

Beaucoup de juristes dans la littérature *AI and Law*, venant de l'école formaliste en droit qui s'oppose à l'école réaliste.

$\Rightarrow$ D'abord, comprendre le contexte d'utilisation des algorithmes en droit

Co-conception avec les juristes

Beaucoup de juristes dans la littérature *AI and Law*, venant de l'école formaliste en droit qui s'oppose à l'école réaliste.

⇒ D'abord, comprendre le contexte d'utilisation des algorithmes en droit

Pendant la conception du langage

- syntaxe choisie par les juristes (sur proposition des informaticiens)
- évaluation sur des exemples choisis par les juristes
- confrontation avec des philosophes du droit

Déployer Catala pour la production

Interprétation ✗

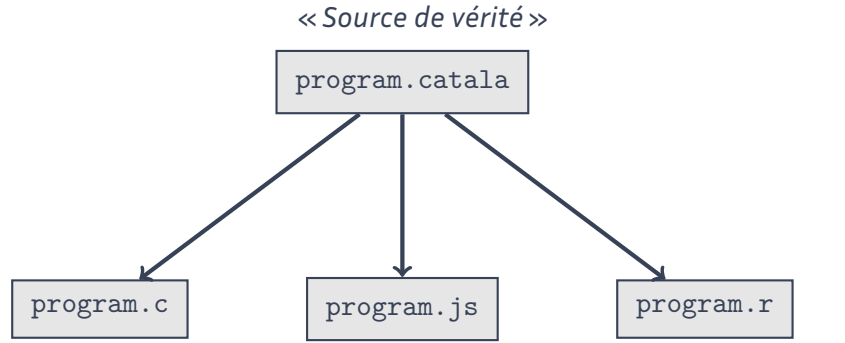
Déployer Catala pour la production

Interprétation ✗

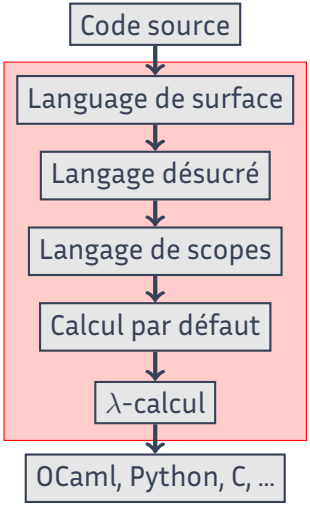
Compilation ✓

Déployer Catala pour la production

Interprétation ✗ Compilation ✓

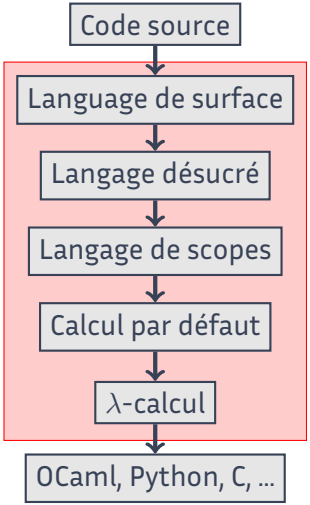


Le compilateur de Catala



- **Architecture en multiples passes**

Le compilateur de Catala



- **Architecture en multiples passes**
- 13 000 lines of OCaml
- Licence Apache2
- Stage M2/M1 à pourvoir au printemps 2025 !
- `https://github.com/CatalaLang/Catala`

Conclusion

Design de language orienté vers la preuve et centré sur l'utilisateur

Design de language orienté vers la preuve et centré sur l'utilisateur

Parti pris

Design de language orienté vers la preuve et centré sur l'utilisateur

Parti pris

La recherche appliquée en méthodes formelles s'est trop concentrée sur les problèmes technique, se spécialisant dans des niches (cryptographie, kernels, etc.) à rendements décroissants.

Design de language orienté vers la preuve et centré sur l'utilisateur

Parti pris

La recherche appliquée en méthodes formelles s'est trop concentrée sur les problèmes technique, se spécialisant dans des niches (cryptographie, kernels, etc.) à rendements décroissants.

PLIERS, User-centered language design [Cob+21] :

> *« Tard dans le projet, nous nous sommes rendus compte que concevoir et faire les études utilisateur de fonctionnalités de bas niveau demandaient typiquement plus de temps que d'implémenter ces fonctionnalités. »*

References

References I

[Bla08] Bruno Blanchet. **« A computationally sound mechanized prover for security protocols ».** In : *IEEE Transactions on Dependable and Secure Computing* 5.4 (2008), p. 193-207.

[Bla16] Bruno Blanchet. **« Modeling and verifying security protocols with the applied pi-calculus and ProVerif ».** In : *Foundations and Trends® in Privacy and Security* 1.1-2 (2016), p. 1-135.

References II

[Chl+17] Adam Chlipala, Benjamin Delaware, Samuel Duchovni, Jason Gross, Clément Pit-Claudel, Sorawit Suriyakarn, Peng Wang et Katherine Ye. **« The end of history ? Using a proof assistant to replace language design with library design ».** In : *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2017.

[Chl13] Adam Chlipala. **« The Bedrock structured programming system : Combining generative metaprogramming and Hoare logic in an extensible program verifier ».** In : *Proceedings of the 18th ACM SIGPLAN international conference on Functional programming*. 2013, p. 391-402.

References III

[Cob+21] Michael Coblenz, Gauri Kambhatla, Paulette Koronkevich, Jenna L Wise, Celeste Barnaby, Joshua Sunshine, Jonathan Aldrich et Brad A Myers. **« PLIERS : A Process that Integrates User-Centered Methods into Programming Language Design ».** In : *ACM Transactions on Computer-Human Interaction (TOCHI)* 28.4 (2021), p. 1-53.

[Del+15] Benjamin Delaware, Clément Pit-Claudel, Jason Gross et Adam Chlipala. **« Fiat : Deductive synthesis of abstract data types in a proof assistant ».** In : *Acm Sigplan Notices* 50.1 (2015), p. 689-700.

References IV

[Fro+21] Aymeric Fromherz, Aseem Rastogi, Nikhil Swamy, Sydney Gibson, Guido Martinez, Denis Merigoux et Tahina Ramananandro. **« Steel : Proof-Oriented Programming in a Dependently Typed Concurrent Separation Logic ».** In : *Proc. ACM Program. Lang.* 5.ICFP (août 2021). doi : 10.1145/3473590. url : https://doi.org/10.1145/3473590.

[Jun+18] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal et Derek Dreyer. **« Iris from the ground up : A modular foundation for higher-order concurrent separation logic ».** In : *Journal of Functional Programming* 28 (2018).

[Law18] Sarah B. Lawsky. **« A Logic for Statutes ».** In : *Florida Tax Review* (2018).

References V

[Lei10] K Rustan M Leino. **« Dafny : An automatic program verifier for functional correctness ».** In : *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer. 2010, p. 348-370.

[Ler06] Xavier Leroy. **« Formal Certification of a Compiler Back-end or : Programming a Compiler with a Proof Assistant ».** In : *SIGPLAN Not.* 41.1 (jan. 2006), p. 42-54.

[MCP21] Denis Merigoux, Nicolas Chataing et Jonathan Protzenko. **« Catala : A Programming Language for the Law ».** In : *Proc. ACM Program. Lang.* 5.ICFP (août 2021). doi : 10.1145/3473582. url : https://doi.org/10.1145/3473582.

References VI

[MKB21] Denis Merigoux, Franziskus Kiefer et Karthikeyan Bhargavan. ***Hacspec : succinct, executable, verifiable specifications for high-assurance cryptography embedded in Rust.*** Technical Report. Inria, mars 2021. url : https://hal.inria.fr/hal-03176482.

[MMP21] Denis Merigoux, Raphaël Monat et Jonathan Protzenko. **« A Modern Compiler for the French Tax Code ».** In : *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*. CC 2021. Virtual, Republic of Korea : Association for Computing Machinery, 2021, p. 71-82. isbn : 9781450383257. doi : 10.1145/3446804.3446850. url : https://doi.org/10.1145/3446804.3446850.

References VII

[Pro+17] Jonathan Protzenko, Jean-Karim Zinzindohoué, Aseem Rastogi, Tahina Ramananandro, Peng Wang, Santiago Zanella-Béguelin, Antoine Delignat-Lavaud, Catalin Hritcu, Karthikeyan Bhargavan, Cédric Fournet et Nikhil Swamy. **« Verified Low-Level Programming Embedded in F* ».** In : *PACMPL* 1.ICFP (sept. 2017), 17 :1-17 :29. doi : 10.1145/3110261. url : http://arxiv.org/abs/1703.00053.

[Pro+19] Jonathan Protzenko, Benjamin Beurdouche, Denis Merigoux et Karthikeyan Bhargavan. **« Formally Verified Cryptographic Web Applications in WebAssembly ».** In : *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, p. 1256-1274. doi : 10.1109/SP.2019.00064.

References VIII

[Swa+16] Nikhil Swamy, Cătălin Hriţcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss et al. **« Dependent types and multi-monadic effects in F* ».** In : *Proceedings of the 43rd annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 2016, p. 256-270.

[Swa+20] Nikhil Swamy, Aseem Rastogi, Aymeric Fromherz, Denis Merigoux, Danel Ahman et Guido Martinez. **« SteelCore : An Extensible Concurrent Separation Logic for Effectful Dependently Typed Programs ».** In : *Proc. ACM Program. Lang.* 4.ICFP (août 2020). doi : 10.1145/3409003. url : https://doi.org/10.1145/3409003.

References IX

[Tuc08] Harvey Tuch. **« Formal memory models for verifying C systems code ».** Thèse de doct. 2008.

[Zim19] Théo Zimmermann. **« Challenges in the collaborative evolution of a proof language and its ecosystem ».** Theses. Université de Paris, déc. 2019. url : `https://hal.inria.fr/tel-02451322`.