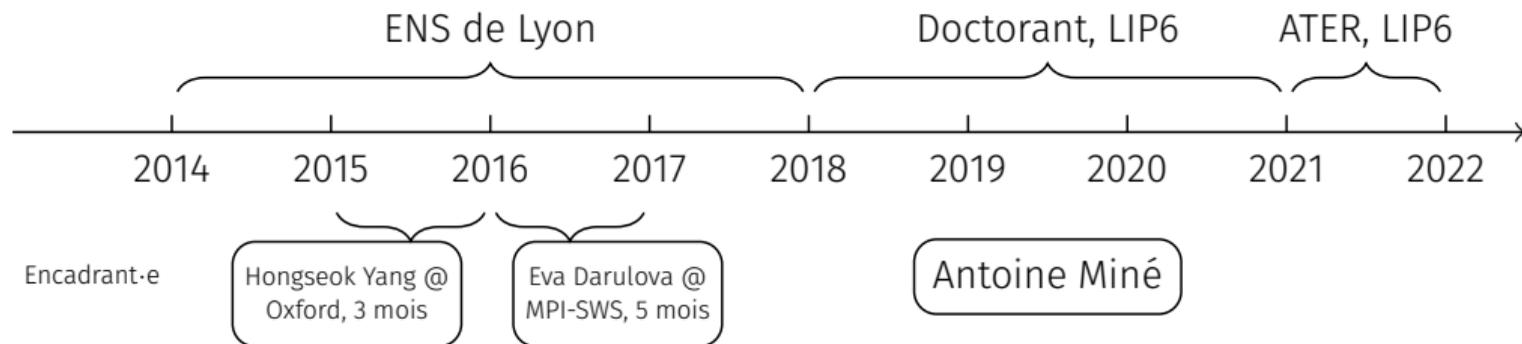


# Candidature MCF ENS de Lyon

## Raphaël Monat – intégration dans l'équipe CASH



# Activités antérieures

---

Contexte

**But des méthodes formelles**

Améliorer la confiance en les logiciels

## But des méthodes formelles

Améliorer la confiance en les logiciels

## Moyens

- ▶ Théorique : définition rigoureuse des systèmes étudiés (sémantique), nouvelles méthodes de preuve, ...
- ▶ Pratique : développements de logiciels

## But des méthodes formelles

Améliorer la confiance en les logiciels

## Moyens

- ▶ Théorique : définition rigoureuse des systèmes étudiés (sémantique), nouvelles méthodes de preuve, ...
- ▶ Pratique : développements de logiciels

## Approche personnelle

Théorie  $\leftrightarrow$  Pratique:

- 1 Trouver des propriétés/programmes concrets
- 2 Étude théorique et développement d'une approche
- 3 Implémentation et validation expérimentale (sur 1)

## But des méthodes formelles

Améliorer la confiance en les logiciels

## Moyens

► Théorique : définitions

► Pratique : de

**Travaux sur deux systèmes réels à grande échelle**

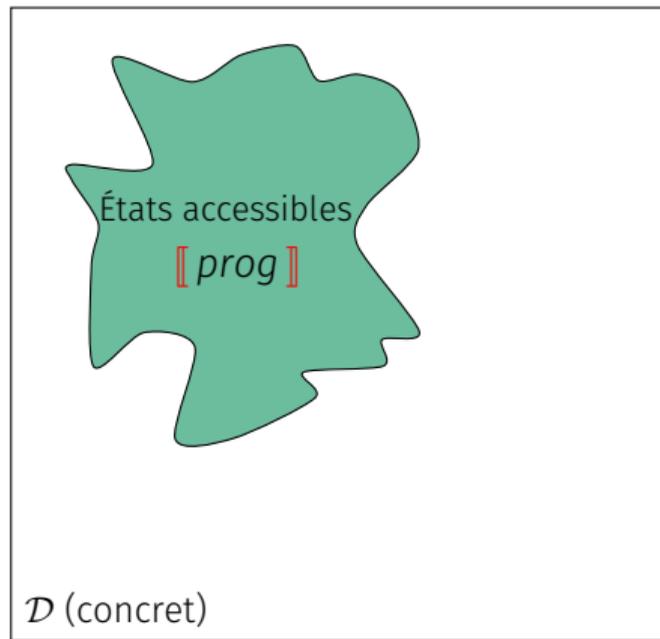
► Python, et mécanisme d'interopérabilité C (LIP6)

► Code de calcul de l'impôt sur le revenu (Inria, MSR)

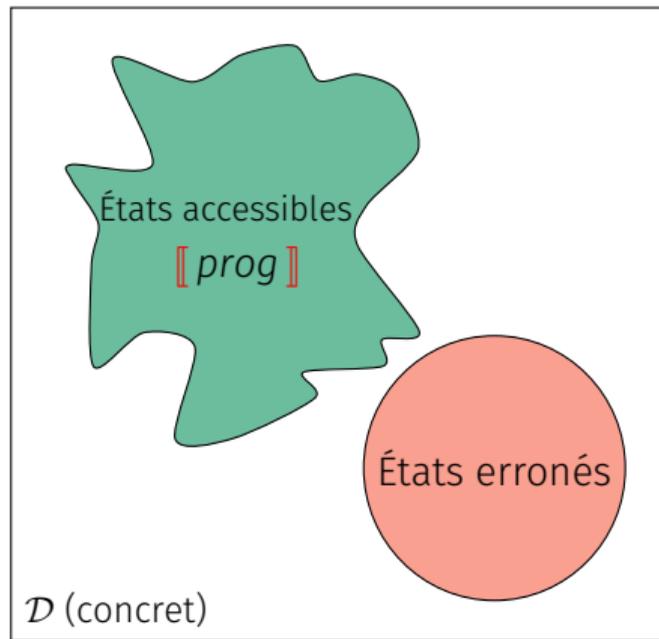
## Approche perso

Théorie  $\leftrightarrow$  Pratique:

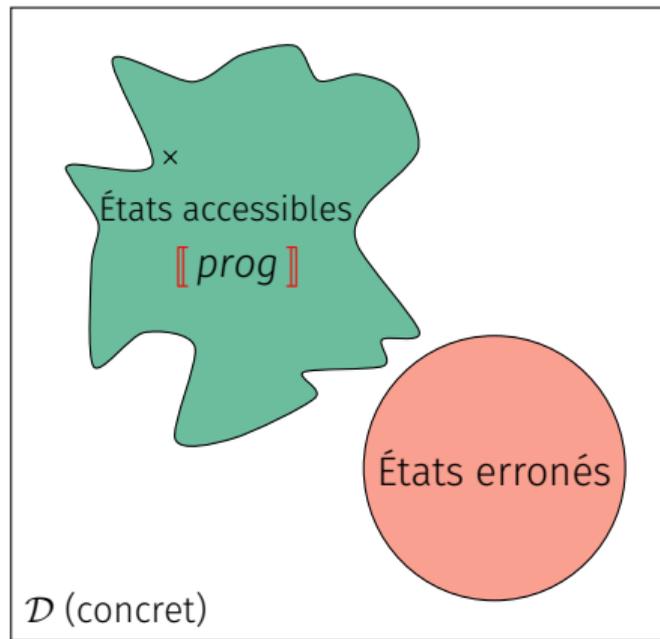
- 1 Trouver des propriétés/programmes concrets
- 2 Étude théorique et développement d'une approche
- 3 Implémentation et validation expérimentale (sur 1)



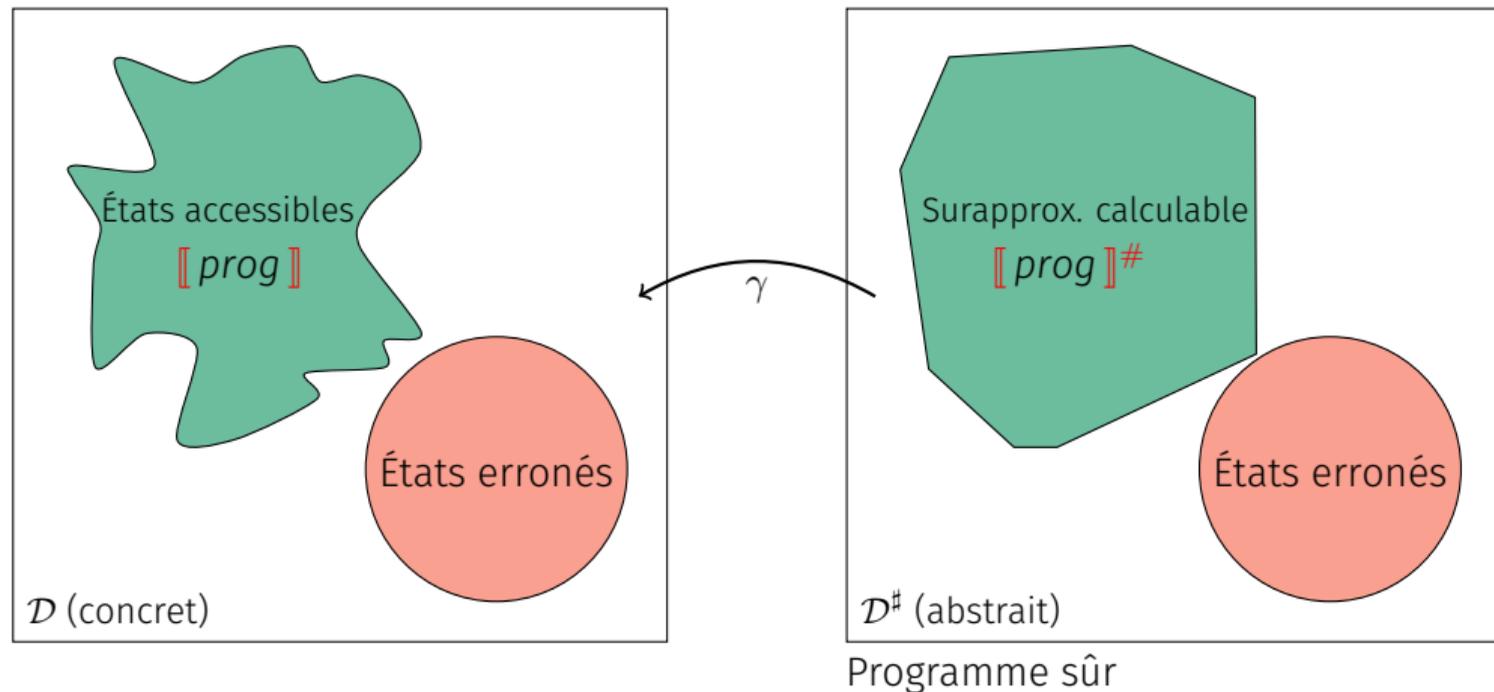
# Analyse statique de programmes par interprétation abstraite



# Analyse statique de programmes par interprétation abstraite

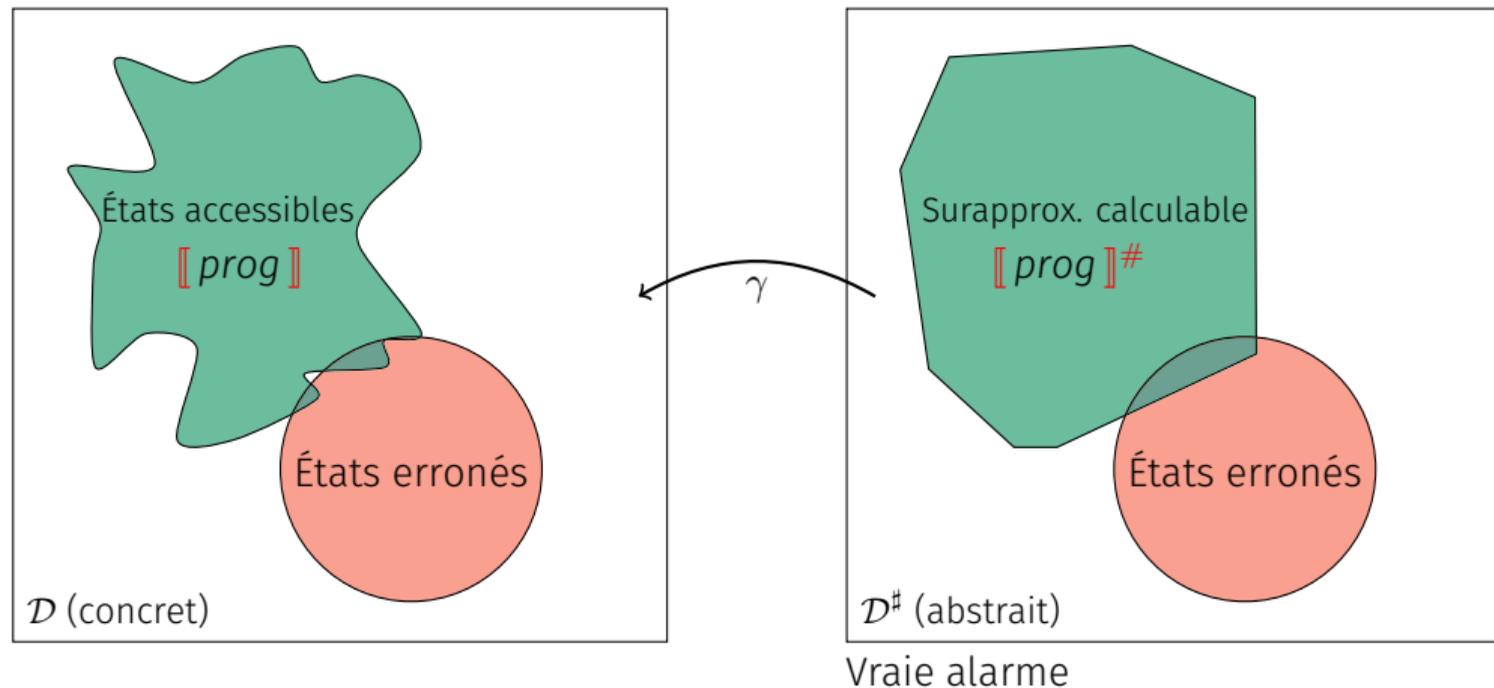


# Analyse statique de programmes par interprétation abstraite



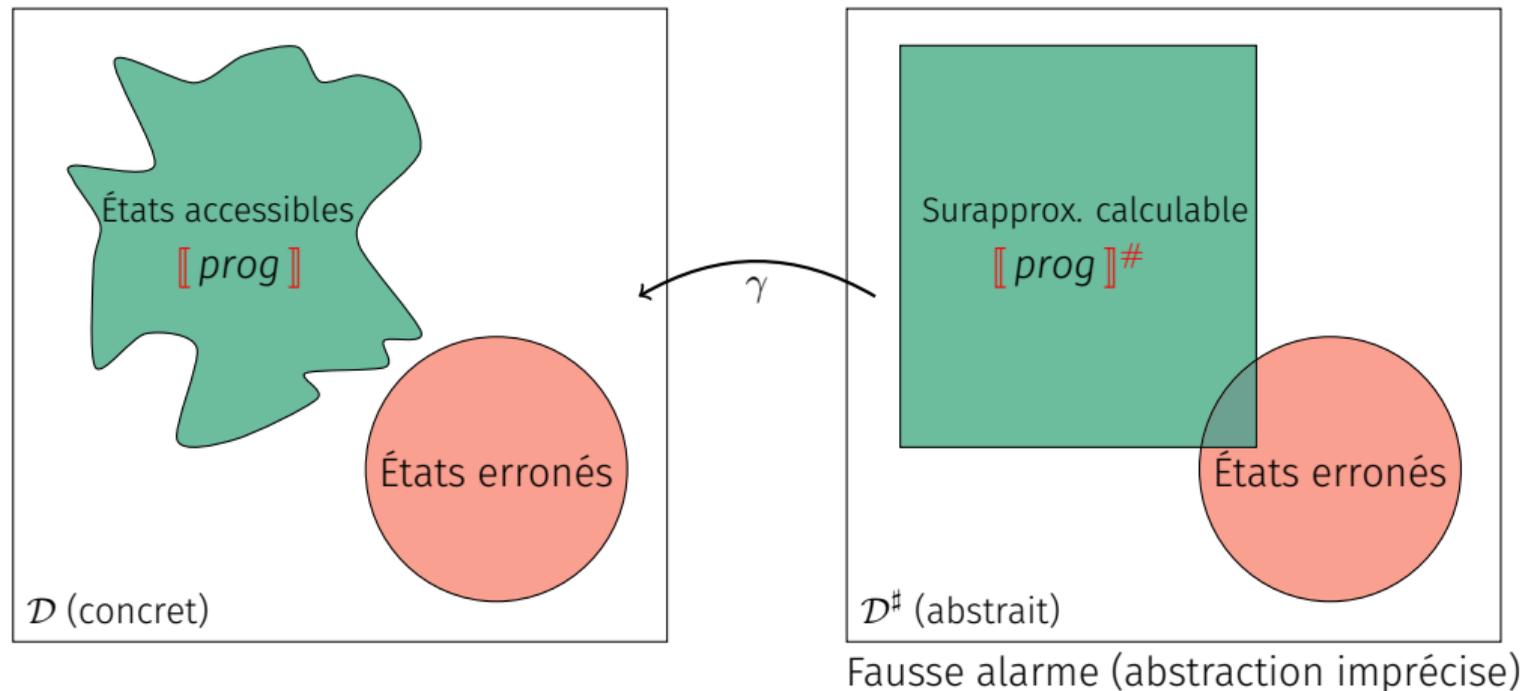
P. Cousot and R. Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". POPL 1977

# Analyse statique de programmes par interprétation abstraite



P. Cousot and R. Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". POPL 1977

# Analyse statique de programmes par interprétation abstraite



P. Cousot and R. Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". POPL 1977

# Certification de programmes critiques par analyse statique



P. Cousot, R. Cousot, Feret, Mauborgne, Miné, Monniaux, and Rival. "Combination of Abstractions in the Astrée Static Analyzer". ASIAN 2006

# Certification de programmes critiques par analyse statique



## C critique

- ▶ Code généré
- ▶ Allocation dynamique

P. Cousot, R. Cousot, Feret, Mauborgne, Miné, Monniaux, and Rival. "Combination of Abstractions in the Astrée Static Analyzer". ASIAN 2006

# Certification de programmes critiques par analyse statique



## C critique

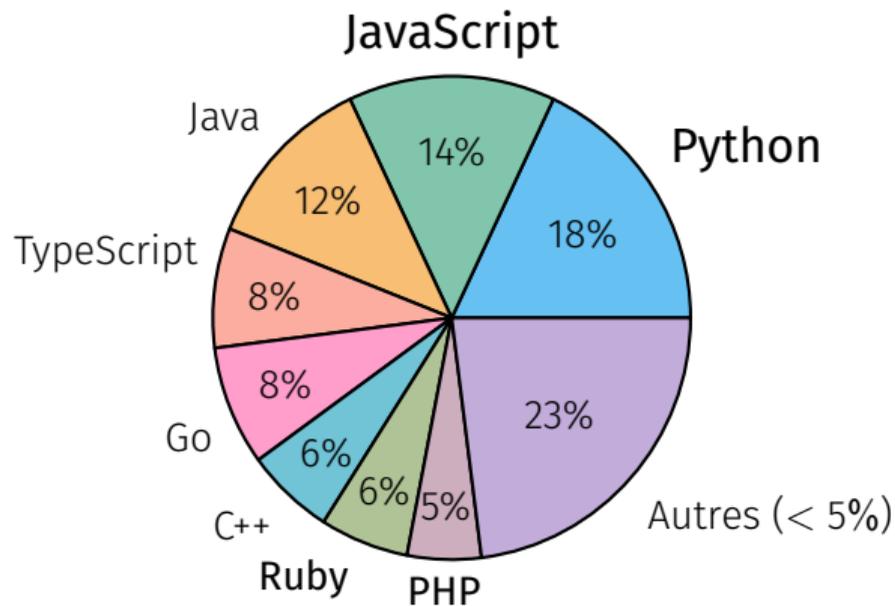
- ▶ Code généré
- ▶ Allocation dynamique

## Comment démocratiser cette approche ?

- ▶ Programmes non critiques
- ▶ Langages de programmation : Python, C, ...
- ▶ Cadre d'analyse générique (langages)

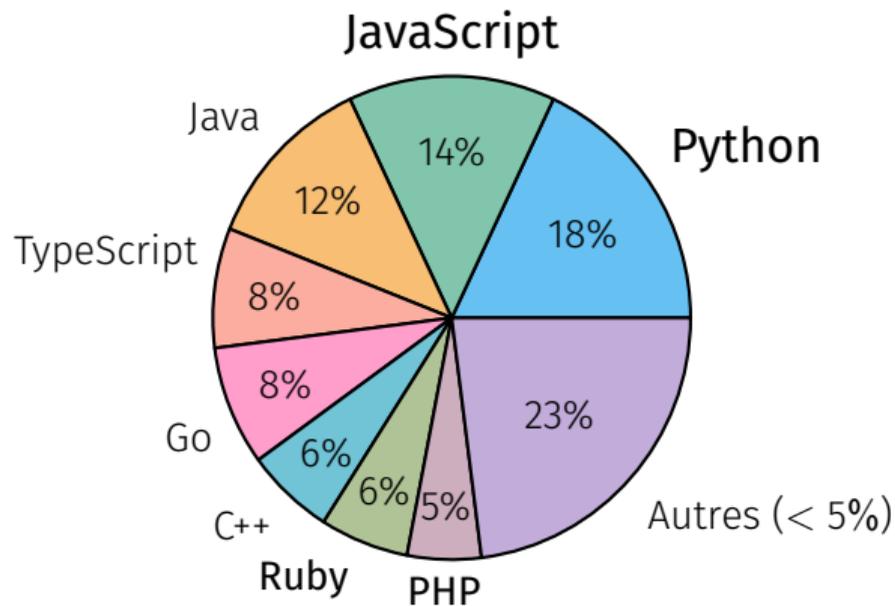
P. Cousot, R. Cousot, Feret, Mauborgne, Miné, Monniaux, and Rival. "Combination of Abstractions in the Astrée Static Analyzer". ASIAN 2006

# Langages de programmation dynamiques (Python, JavaScript, ...)



Langages les plus populaires sur GitHub

# Langages de programmation dynamiques (Python, JavaScript, ...)



Langages les plus populaires sur GitHub

## Particularités

- ▶ Typage dynamique
- ▶ Structure dynamique d'objet

## Activités antérieures

---

Analyse statique de programmes Python & C

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup> $[·]_{py} \rightsquigarrow [·]_{py}^{\#}$

- ▶ Combinaison de nombreuses abstractions
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ Première analyse réellement multilingage
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Ouadjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020

<sup>2</sup>Monat, Ouadjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Ouadjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021

<sup>4</sup>Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup>

- ▶ **Combinaison de nombreuses abstractions**
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ Première analyse réellement multilingage
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Ouadjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020 

<sup>2</sup>Monat, Ouadjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Ouadjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021 

<sup>4</sup>Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup> ["·"]\_py $\rightsquigarrow$ ["·"]\_py#

- ▶ **Combinaison de nombreuses abstractions**
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ Première analyse réellement multilingage
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Ouadjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020

<sup>2</sup>Monat, Ouadjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Ouadjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021

<sup>4</sup>Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup> `[·]_py` $\rightsquigarrow$ `[·]_py^#`

- ▶ **Combinaison de nombreuses abstractions**
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ **Première analyse réellement multilingage**
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Ouadjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020

<sup>2</sup>Monat, Ouadjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Ouadjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021

<sup>4</sup>Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup> []\_py $\rightsquigarrow$ []\_py#

- ▶ **Combinaison de nombreuses abstractions**
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ **Première analyse réellement multilingage**
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Oudjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020

<sup>2</sup>Monat, Oudjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Oudjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021

<sup>4</sup>Journault, Miné, Monat, and Oudjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

# Contributions autour de l'analyse de programmes Python

## Sémantique de Python



- ▶ Rétro-ingénierie depuis CPython (160kLoc C)
- ▶ Lien avec le code source (auditabilité)
- ▶ Formalisation dans manuscrit ( $\simeq$  44 pages)

## Analyses de type et de valeur<sup>1,2</sup>

- ▶ Combinaison de nombreuses abstractions
- ▶ Analyse de benchmarks CPython

## Analyse multilingage Python/C<sup>3</sup>

- ▶ Première analyse réellement multilingage
- ▶ Analyse de bibliothèques réelles

## Implémentation dans Mopsa<sup>4</sup>

- ▶ Analyseur open-source pour Python et C
- ▶ Factorisation d'abstractions entre langages
- ▶ Partage entre états des abstractions

<sup>1</sup>Monat, Ouadjaout, and Miné. "Static Type Analysis by Abstract Interpretation of Python Programs". ECOOP 2020

<sup>2</sup>Monat, Ouadjaout, and Miné. "Value and allocation sensitivity in static Python analyses". SOAP@PLDI 2020 (workshop)

<sup>3</sup>Monat, Ouadjaout, and Miné. "A Multilanguage Static Analysis of Python Programs with Native C Extensions". SAS 2021

<sup>4</sup>Journault, Miné, Monat, and Ouadjaout. "Combinations of reusable abstract domains for a multilingual static analyzer". VSTTE 2019 (invité)

Constat : 20% des 200 bibliothèques Python les plus populaires contiennent du C

- ▶ Performance (numpy)
- ▶ Bibliothèques systèmes (pygit2)

Constat : 20% des 200 bibliothèques Python les plus populaires contiennent du C

- ▶ Performance (numpy)
- ▶ Bibliothèques systèmes (pygit2)

### Dangers

- ▶ Valeurs différentes ( $\mathbb{Z}$  vs. `Int32`)
- ▶ Partage de l'état mémoire

Constat : 20% des 200 bibliothèques Python les plus populaires contiennent du C

- ▶ Performance (numpy)
- ▶ Bibliothèques systèmes (pygit2)

## Dangers

- ▶ Valeurs différentes ( $\mathbb{Z}$  vs. `Int32`)
- ▶ Partage de l'état mémoire

## Notre approche

- ▶ Analyse **combinée** du code C, Python, et de l'interface
- ▶ Travaux précédents<sup>5</sup> : JNI  Java, peu précis

<sup>5</sup>Tan and Morrisett. "Ilea: inter-language analysis across Java and C". OOPSLA 2007; Furr and Foster. "Checking type safety of foreign function calls". 2008; Lee, Lee, and Ryu. "Broadening Horizons of Multilingual Static Analysis: Semantic Summary Extraction from C Code for JNI Program Analysis". ASE 2020

### Difficulté : partage de l'état mémoire

- ▶ Deux visions distinctes d'un état mémoire partagé
- ▶ Synchronisation ?
  - Traduction complète coûteuse
  - Mopsa permet de partager certaines abstractions

## Difficulté : partage de l'état mémoire

- ▶ Deux visions distinctes d'un état mémoire partagé
- ▶ Synchronisation ?
  - Traduction complète coûteuse
  - Mopsa permet de partager certaines abstractions

## Séparation de l'état et synchronisation réduite

Observation : structures directement déréréférencables par un seul langage.

## Difficulté : partage de l'état mémoire

- ▶ Deux visions distinctes d'un état mémoire partagé
- ▶ Synchronisation ?
  - Traduction complète coûteuse
  - Mopsa permet de partager certaines abstractions

## Séparation de l'état et synchronisation réduite

Observation : structures directement déréréférencables par un seul langage. Python accède au C via des accesseurs (et vice-versa).

## Difficulté : partage de l'état mémoire

- ▶ Deux visions distinctes d'un état mémoire partagé
- ▶ Synchronisation ?
  - Traduction complète coûteuse
  - Mopsa permet de partager certaines abstractions

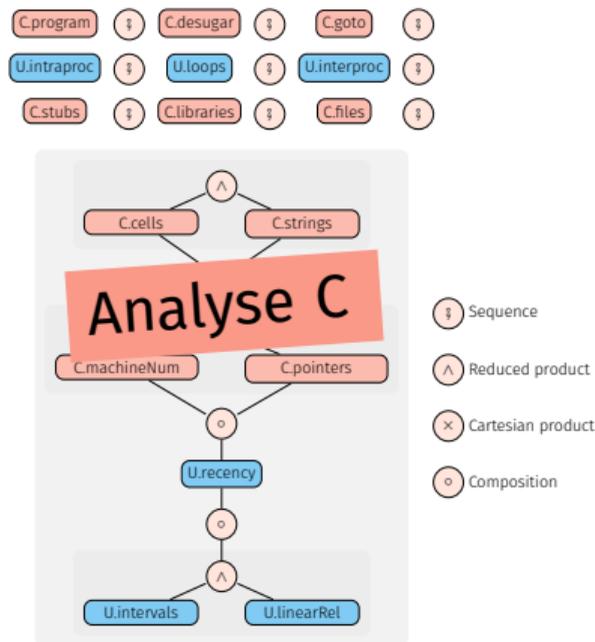
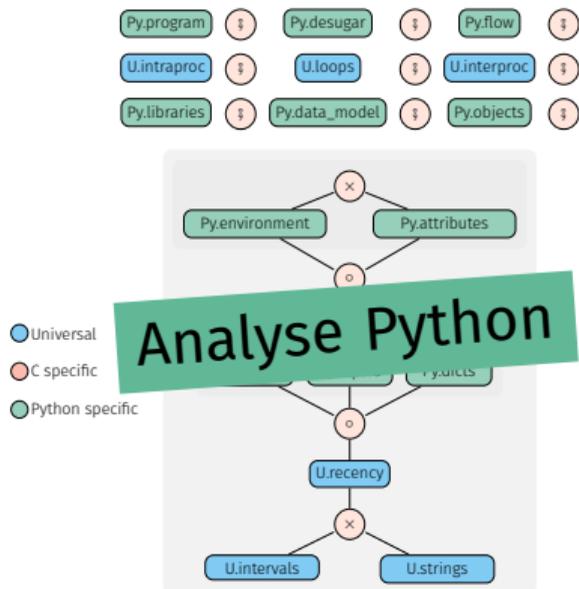
## Séparation de l'état et synchronisation réduite

Observation : structures directement déréférencables par un seul langage. Python accède au C via des accesseurs (et vice-versa).

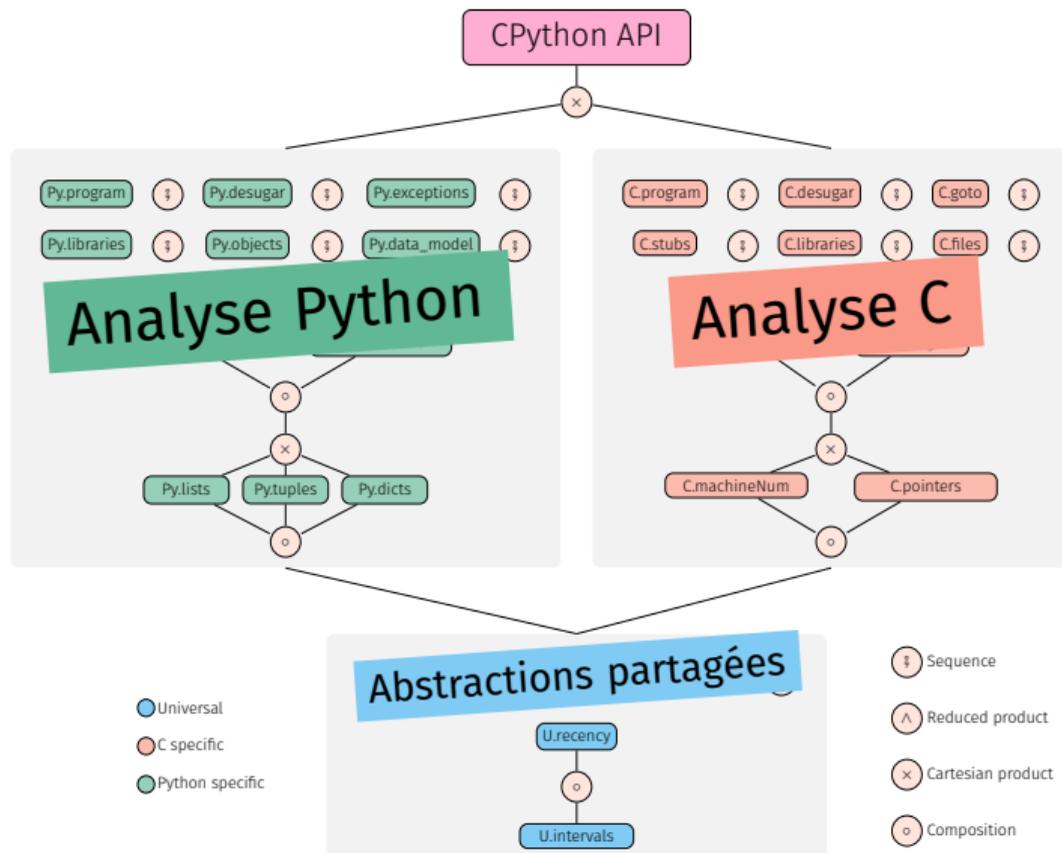
⇒ Séparation relative de l'état

- ▶ Hypothèse : accès C aux structures Python via API
- ▶ Synchronisation réduite : lorsqu'un objet change de langage pour la première fois

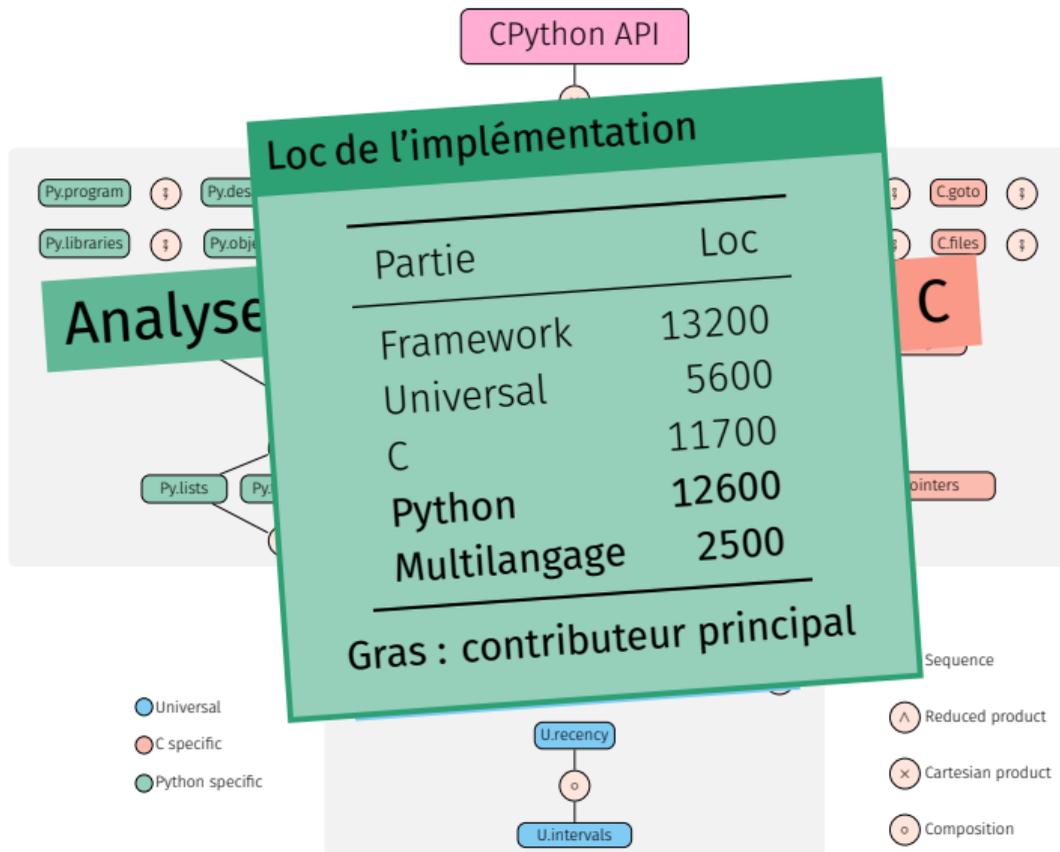
# Analyse multilangage – implémentation



# Analyse multilangage – implémentation



# Analyse multilangage – implémentation



## Sélection du corpus

- ▶ Bibliothèques populaires disponibles sur GitHub, moyenne 412 étoiles
- ▶ Analyse de programmes complets : tests comme code client

Bibliothèque	C + Py. Loc	Tests	🕒/test	$\frac{\# \text{ vérif. prouvées}}{\# \text{ vérif.}} \%$	# vérif.
noise	1397	15/15	1.2s	99.7%	(6690)
cdistance	2345	28/28	4.1s	98.0%	(13716)
l1ist	4515	167/194	1.5s	98.8%	(36255)
ahocorasick	4877	46/92	1.2s	96.7%	(6722)
levenshtein	5798	17/17	5.3s	84.6%	(4825)
bitarray	5841	159/216	1.6s	94.9%	(25566)

## Sélection du corpus

- ▶ Bibliothèques populaires disponibles sur GitHub, moyenne 412 étoiles
- ▶ Analyse de programmes complets : tests comme code client

Bibliothèque	C + Py. Loc	Tests	🕒/test	$\frac{\# \text{ vérif. prouvées}}{\# \text{ vérif.}} \%$	# vérif.
noise	1397	15/15	1.2s	99.7%	(6690)
cdistance	2345	28/28	4.1s	98.0%	(13716)
l1ist	4515	167/194	1.5s	98.8%	(36255)
ahocorasick	4877	46/92	1.2s	96.7%	(6722)
levenshtein	5798	17/17	5.3s	84.6%	(4825)
<b>bitarray</b>	<b>5841</b>	159/216	<b>1.6s</b>	<b>94.9%</b>	(25566)

## Activités antérieures

---

Code de calcul de l'impôt sur le revenu

## Impôt sur le revenu des particuliers

- ▶ 38M de foyers fiscaux, 75Md€ de «recettes»
- ▶ Code publié depuis avril 2016 : 92kLoc M, langage spécifique DGFIP
- ⚠ États des lieux en 2019 : calcul non reproductible

## Impôt sur le revenu des particuliers

- ▶ 38M de foyers fiscaux, 75Md€ de «recettes»
- ▶ Code publié depuis avril 2016 : 92kLoc M, langage spécifique DGFIP
- ▶ ⚠ États des lieux en 2019 : calcul non reproductible

## Quelle confiance accorder aux implémentations de codes juridiques ?

- ▶ Reproductibilité des décisions ?
- ▶ Respect de la loi, qui agit comme spécification ?

## Contributions sur le calcul de l'impôt sur le revenu

- ▶ Formalisé      sémantique de M en Coq
- ▶ Reproductible    compilateur open-source MLANG, longuement testé
- ▶ Extensible      6kLoc C  $\rightsquigarrow$  100Loc M++, DSL

## Contributions sur le calcul de l'impôt sur le revenu

- ▶ Formalisé      sémantique de M en Coq
- ▶ Reproductible    compilateur open-source MLANG, longuement testé
- ▶ Extensible      6kLoc C  $\rightsquigarrow$  100Loc M++, DSL

## Communication avec l'administration

- ▶ Travail sur le temps long : 9 mois pour accéder à du code C manquant
- ▶ Pédagogie avec la DGFIP, milieu très juridique

# Contributions

## Contributions sur le calcul de l'impôt sur le revenu

- ▶ Formalisé sémantique de M en Coq
- ▶ Reproductible compilateur open-source MLANG. longuement testé
- ▶ Extensible

**Transfert technologique auprès de la DGFiP !**

- ▶ mission d'expertise de 30 jours de janvier à août 2022
- ▶ encadrement de trois développeurs (OCamlPro & DGFiP)

## Communication

- ▶ Travail sur le temps long : 9 mois pour accéder à du code C manquant
- ▶ Pédagogie avec la DGFiP, milieu très juridique

## Analyse statique de programmes Python utilisant des bibliothèques C

Ouadaout et Miné (LIP6, Sorbonne Université)

- ▶ **SAS'21** , **ECOOP'20** , VSTTE'19 (invité), **SOAP@PLDI'20** , JFLA'21 (fr, outil)
- ▶ Orateur invité au Facebook Testing and Verification Symposium 2021
- ▶ Mopsa (LGPL v3, 60kLoc OCaml), contributeur principal depuis septembre 2018

## Analyse statique de programmes Python utilisant des bibliothèques C

Oudjaout et Miné (LIP6, Sorbonne Université)

- ▶ **SAS'21** , **ECOOP'20** , VSTTE'19 (invité), **SOAP@PLDI'20** , JFLA'21 (fr, outil)
- ▶ Orateur invité au Facebook Testing and Verification Symposium 2021
- ▶ Mopsa (LGPL v3, 60kLoc OCaml), contributeur principal depuis septembre 2018

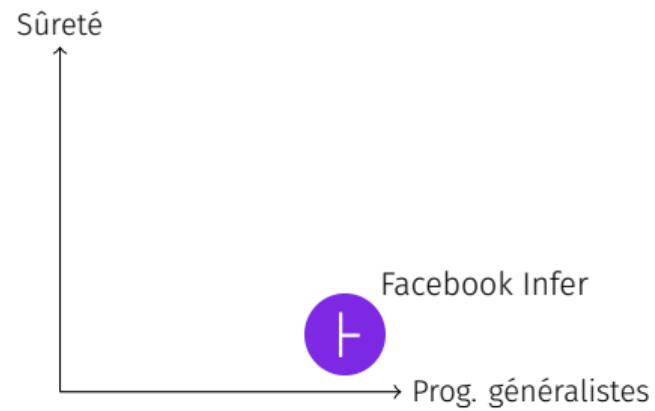
## Code de calcul de l'impôt sur le revenu

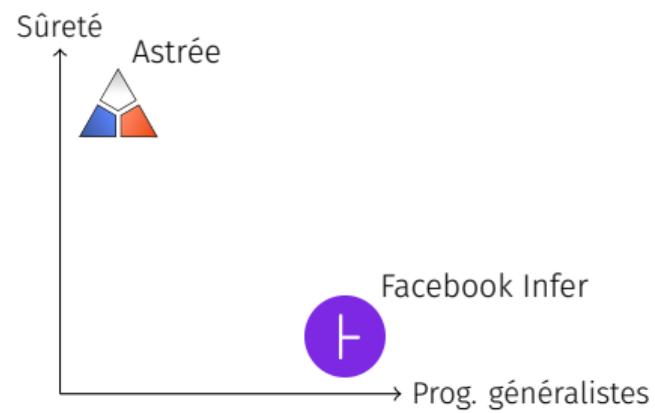
Merigoux (Prosecco, Inria Paris) et Protzenko (Microsoft Research)

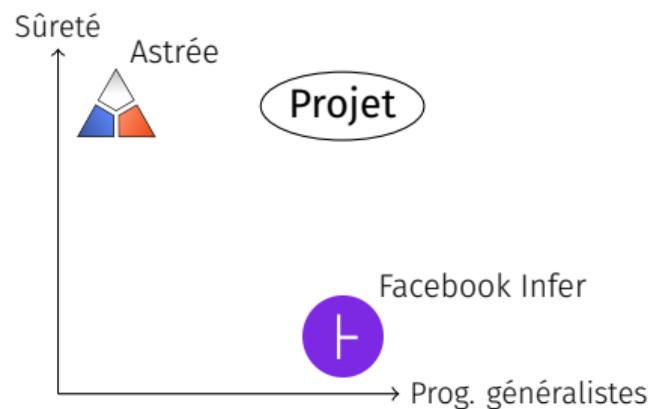
- ▶ **CC'21** , JFLA'20 (fr), JFLA'21 (fr, outil)
- ▶ Compilateur MLANG (GPL v3, 10kLoc OCaml), contributeur principal depuis mai 2019
- ▶ Transfert auprès de la DGFIP : mission d'expertise, encadrement de trois développeurs

# Projet de recherche et intégration

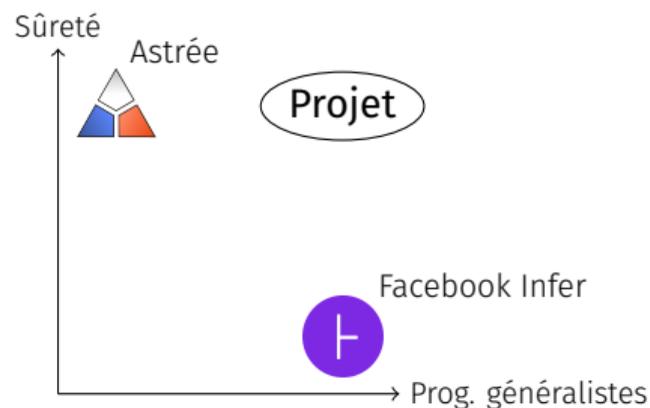
---







⇒ Réunification des approches, pour une adoption massive.



⇒ Réunification des approches, pour une adoption massive.

## Analyses statiques précises, sûres et efficaces pour les logiciels généralistes

- 1 Formalisation et analyse sûre de sémantiques complexes
- 2 Rendre les analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi



- Sémantique formelle exécutable
- ▣ Génération automatique de tests couvrants



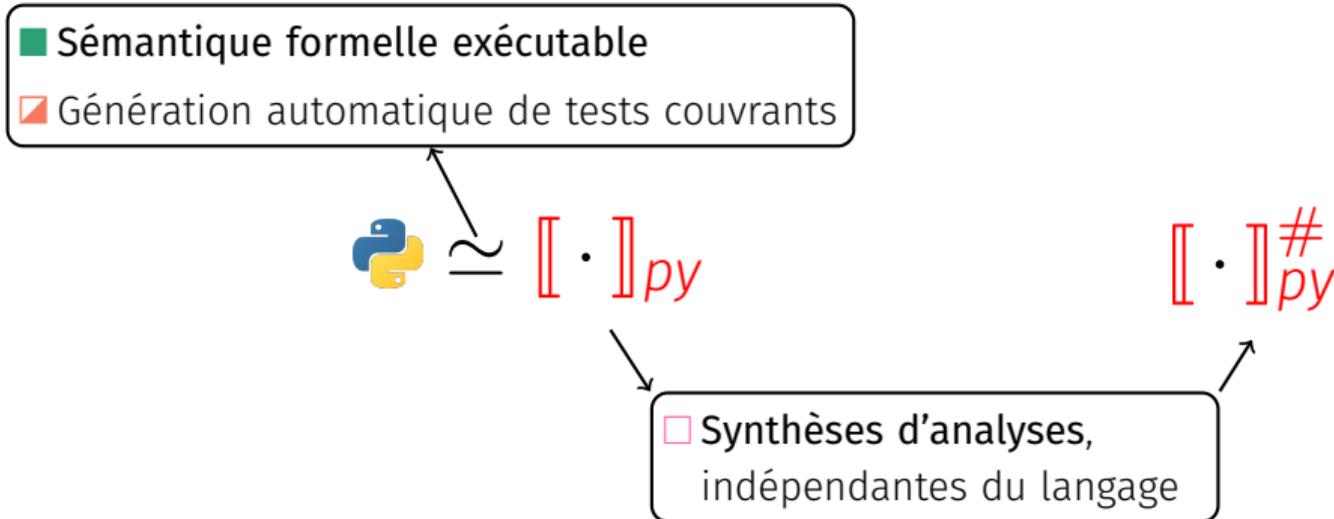
- Sémantique formelle exécutable
- ▣ Génération automatique de tests couvrants

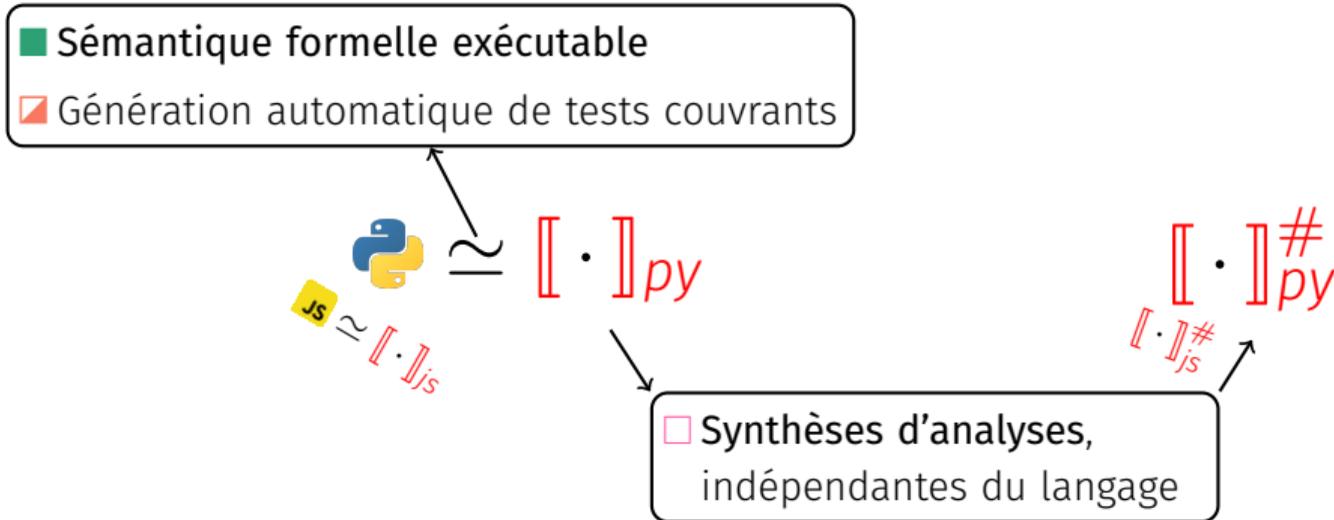


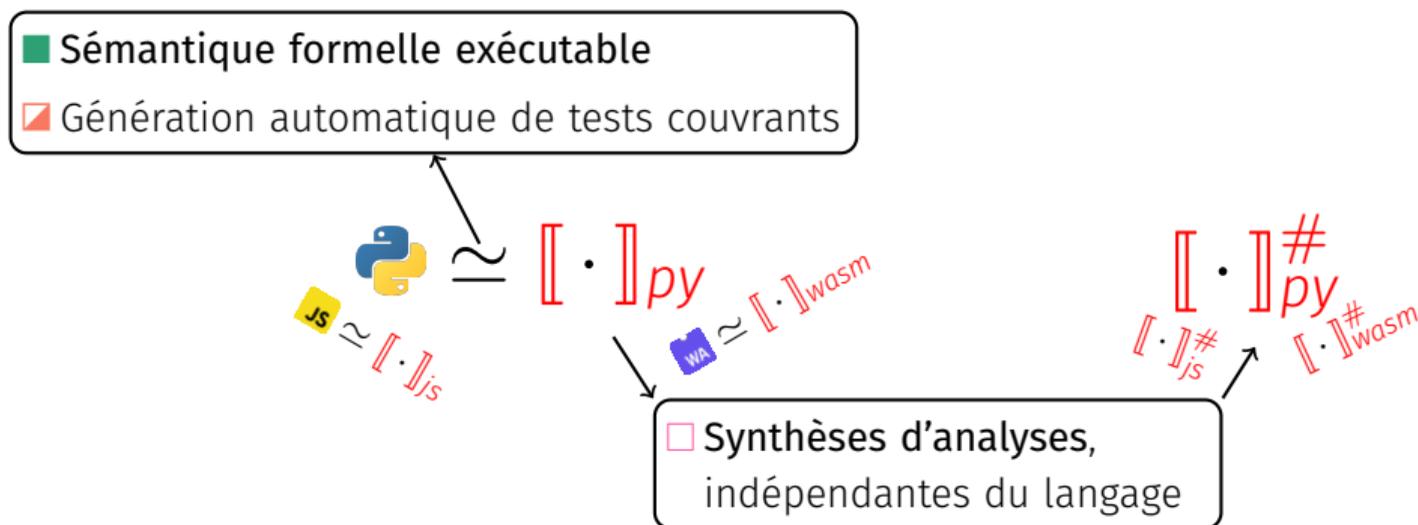
$\approx$

$\llbracket \cdot \rrbracket_{py}$

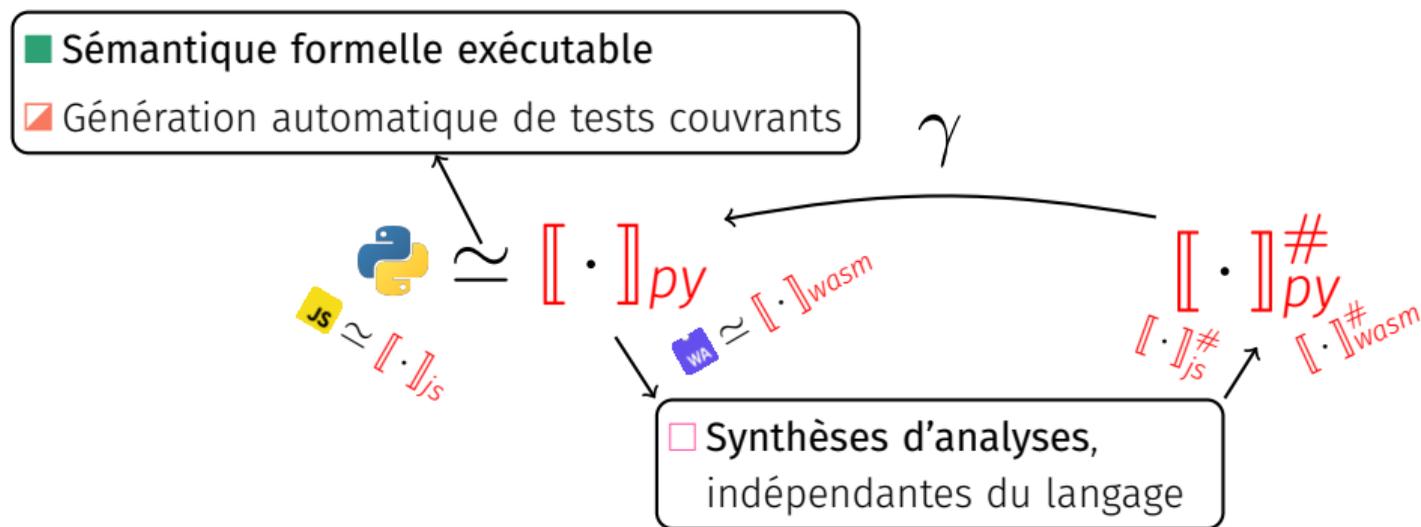
$\llbracket \cdot \rrbracket_{py}^\#$



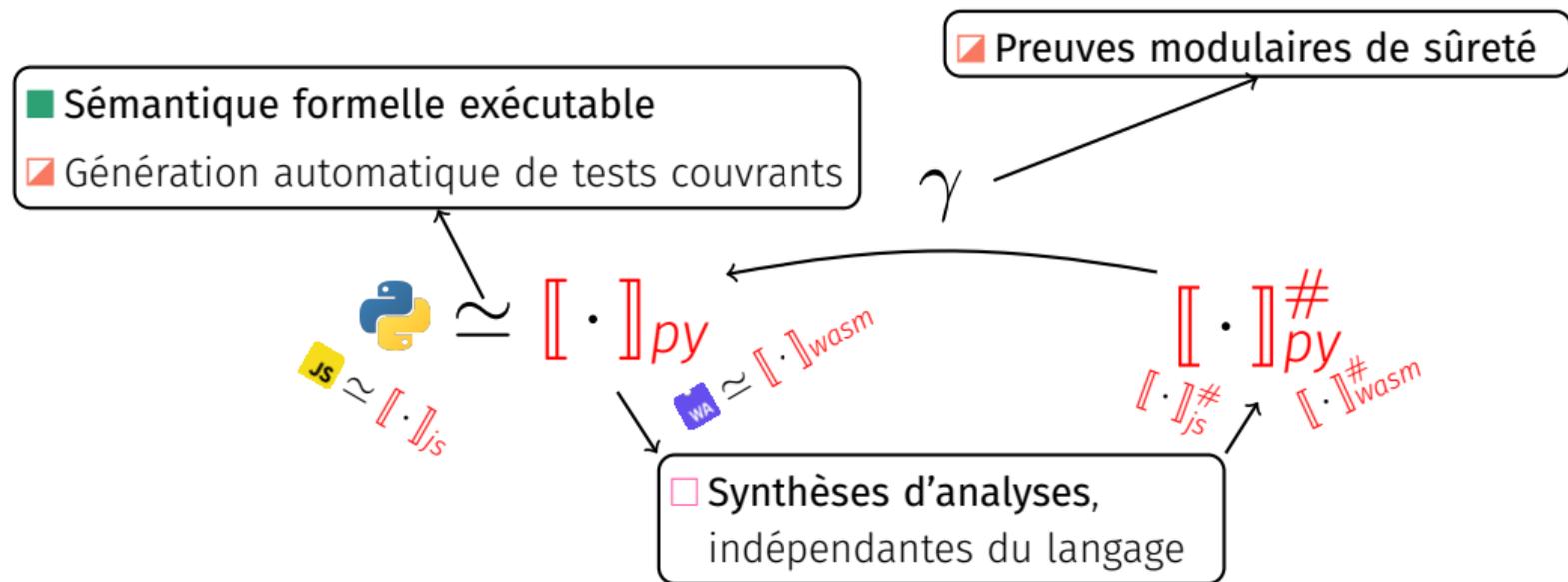




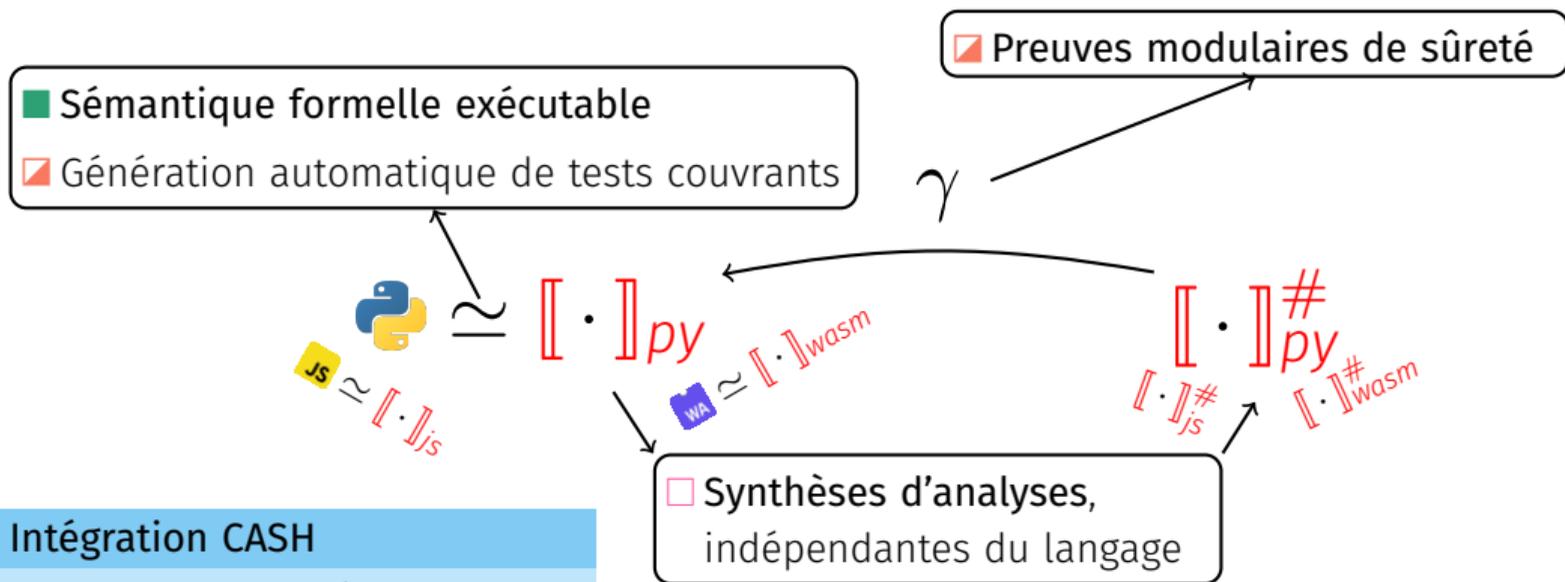
# Axe I – Formalisation et analyse sûre de sémantiques complexes



# Axe I – Formalisation et analyse sûre de sémantiques complexes

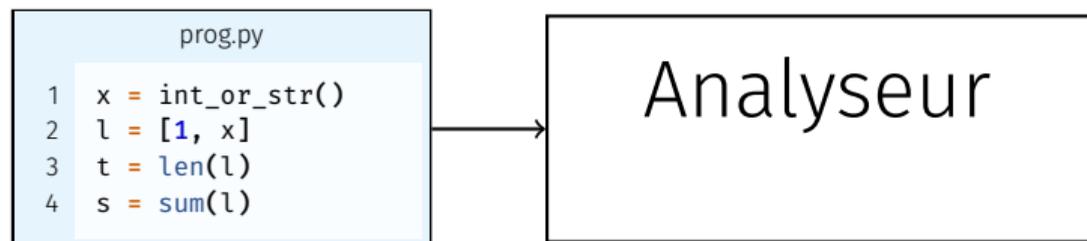


# Axe I – Formalisation et analyse sûre de sémantiques complexes



## Intégration CASH

Zakowski, sem. exécutables  
⇒ Apport expertise sem.  
langages dynamiques



### □ Analyses compositionnelles de fonctions

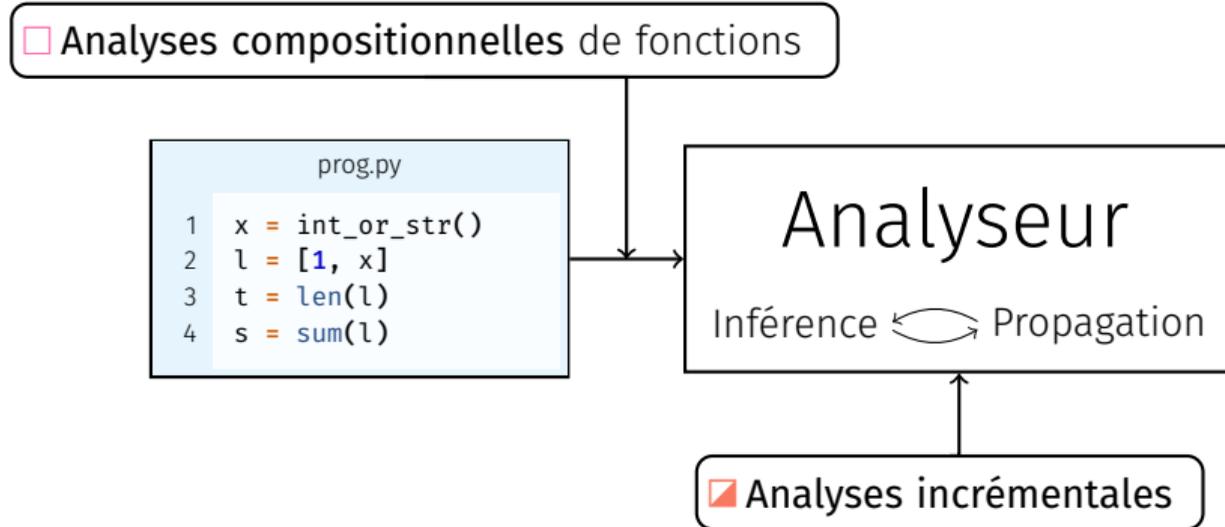
```
prog.py
1 x = int_or_str()
2 l = [1, x]
3 t = len(l)
4 s = sum(l)
```

Analyseur

### □ Analyses compositionnelles de fonctions

```
prog.py
1 x = int_or_str()
2 l = [1, x]
3 t = len(l)
4 s = sum(l)
```





Analyses compositionnelles de fonctions

```
prog.py
1 x = int_or_str()
2 l = [1, x]
3 t = len(l)
4 s = sum(l)
```



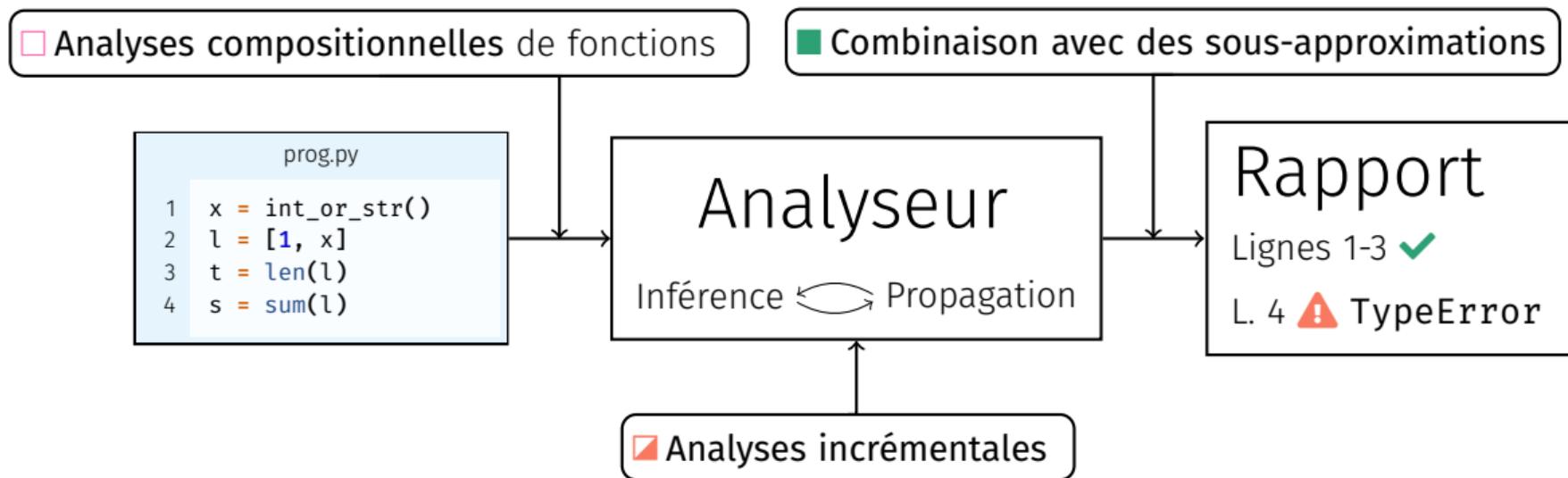
Rapport

Lignes 1-3 ✓

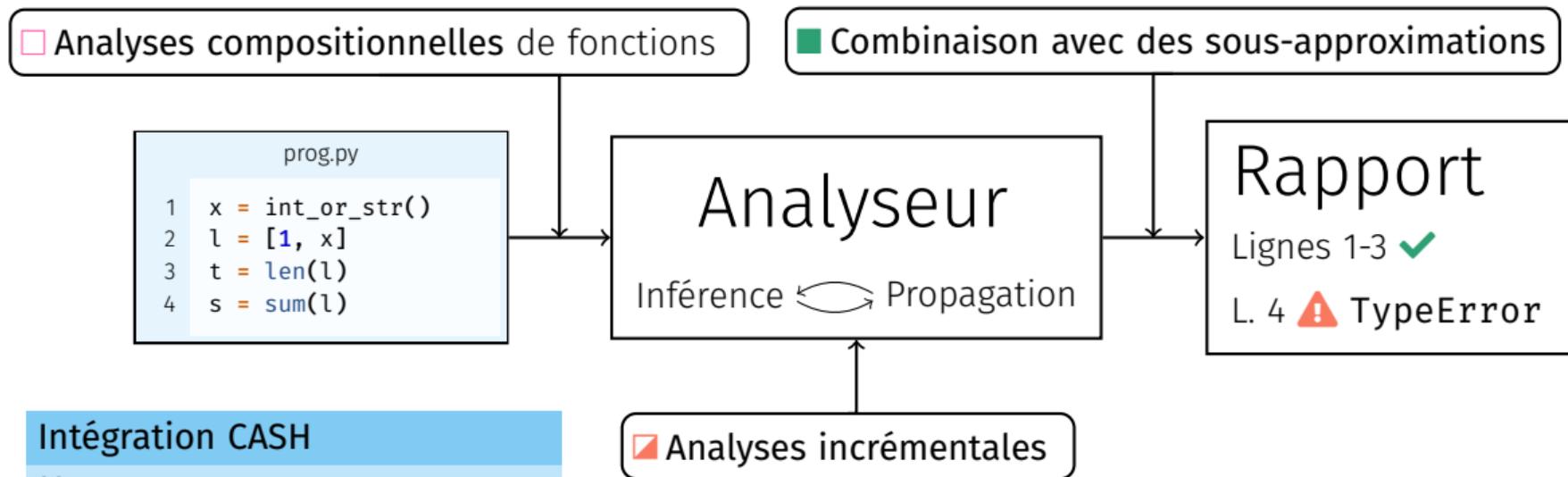
L. 4 ⚠️ TypeError

Analyses incrémentales

## Axe II – Rendre les analyses statiques plus utilisables



## Axe II – Rendre les analyses statiques plus utilisables



### Intégration CASH

Moy

⇒ Renforce pôle interp. abs.

(Gonnord <sup>2021</sup> ⇄ ESISAR)

### L'implémentation de l'impôt sur le revenu est-elle conforme ?

- ▶ Pas de correspondance structurelle
- ▶ 2019~→2020 : 30% des 90kLoc M modifiées

### L'implémentation de l'impôt sur le revenu est-elle conforme ?

- ▶ Pas de correspondance structurelle
- ▶ 2019~→2020 : 30% des 90kLoc M modifiées

### Catala, un nouveau DSL

#### Article D521-1 du code de la sécurité sociale

I - Pour l'application de l'article L 521-1 , le montant des allocations familiales et de la majoration pour âge prévue à l'article L 521-3 est défini selon le barème suivant :

1° Lorsque le ménage ou la personne a disposé d'un montant de ressources inférieur ou égal au plafond défini au I de l'article D. 521-3, les taux servant au calcul des allocations familiales sont fixés, en pourcentage de la base mensuelle prévue à l'article L. 551-1, à :

a) 32 % pour le deuxième enfant à charge ;

```
```catala
```

```
champ d'application AllocationsFamiliales :
```

```
  définition montant_initial_base_deuxième_enfant sous condition
```

```
    ressources_ménage ≤€ plafond_I_d521_3
```

```
  conséquence égal à
```

```
    si nombre de enfants_à_charge_droit_ouvert_prestation_familiale ≥ 2
```

```
    alors prestations_familiales.base_mensuelle ×€ 32 %
```

```
    sinon 0 €
```

```
```
```

### L'implémentation de l'impôt sur le revenu est-elle conforme ?

- ▶ Pas de correspondance structurelle
- ▶ 2019↔2020 : 30% des 90kLoc M modifiées

### Catala, un nouveau DSL

#### Article D521-1 du code de la sécurité sociale

I - Pour l'application de l'article L. 521-1, le montant des allocations familiales et de la majoration pour âge prévue à l'article L. 521-3 est défini selon le barème suivant :

1° Lorsque le ménage ou la personne a disposé d'un montant de ressources inférieur ou égal au plafond défini au I de l'article D. 521-3, les taux servant au calcul des allocations familiales sont fixés, en pourcentage de la base mensuelle prévue à l'article L. 551-1, à :

a) 32 % pour le deuxième enfant à charge ;

```
```catala
```

```
champ d'application AllocationsFamiliales :
```

```
  définition montant_initial_base_deuxième_enfant sous condition  
    ressources_ménage ≤€ plafond_I_d521_3
```

```
  conséquence égal à
```

```
    si nombre de enfants_à_charge_droit_ouvert_prestation_familiale ≥ 2
```

```
      alors prestations_familiales.base_mensuelle ×€ 32 %
```

```
      sinon 0 €
```

```
```
```

- ▶ Programmation littéraire, par binôme juriste/développeur·euse
- ▶ Logique par défaut, pour la loi
- ▶ Participation depuis janvier 2022
  - Formalisation dates/durées
  - Analyse statique de Catala

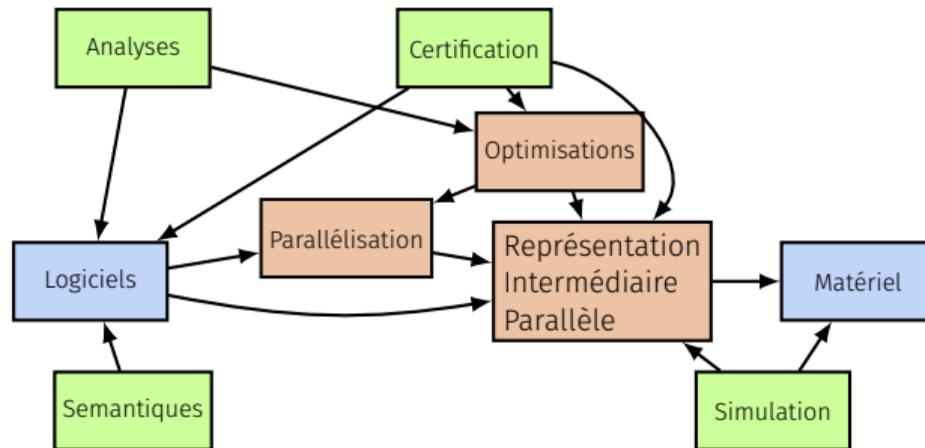


## Projet de recherche

Analyses statiques précises, sûres et efficaces pour les logiciels généralistes

- 1 Formalisation et analyse de sémantiques
- 2 Analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi

# Résumé du projet de recherche – intégration dans l'équipe CASH



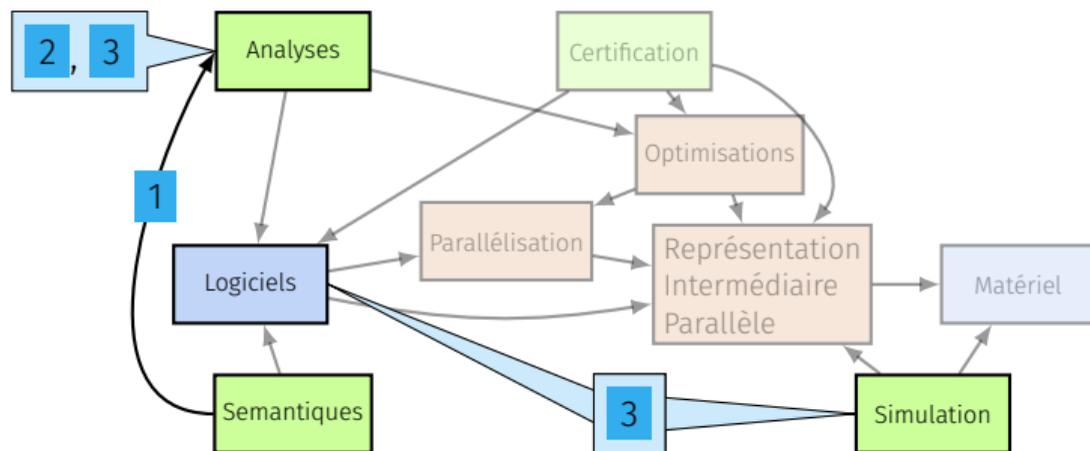
TikZ fourni par l'équipe CASH

## Projet de recherche

Analyses statiques précises, sûres et efficaces pour les logiciels généralistes

- 1 Formalisation et analyse de sémantiques
- 2 Analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi

# Résumé du projet de recherche – intégration dans l'équipe CASH



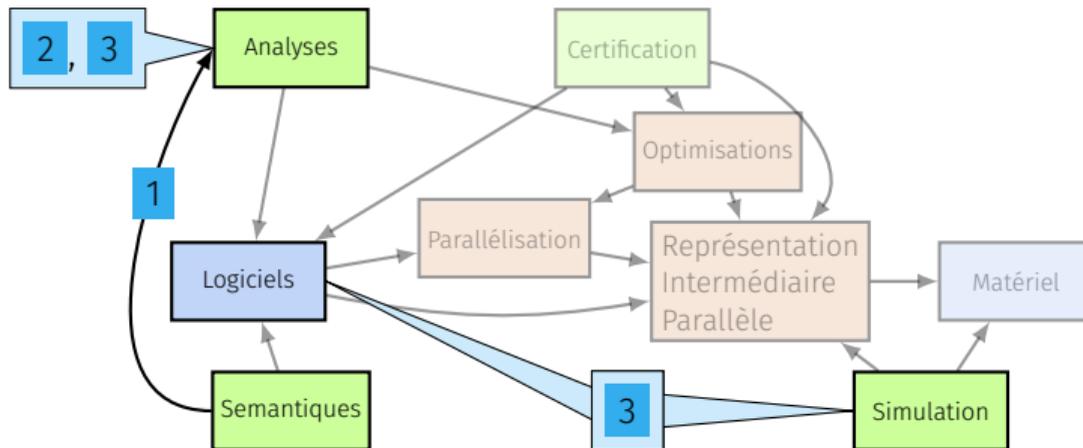
TikZ fourni par l'équipe CASH

## Projet de recherche

Analyses statiques précises, sûres et efficaces pour les logiciels généralistes

- 1 Formalisation et analyse de sémantiques
- 2 Analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi

# Résumé du projet de recherche – intégration dans l'équipe CASH



TikZ fourni par l'équipe CASH

## Projet de recherche

Analyses statiques précises, sûres et efficaces pour les logiciels généralistes

- 1 Formalisation et analyse de sémantiques
- 2 Analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi

## Ancrage local, national, international

- ▶ LIP : Plume, intérêt FP/MF (AriC)
- ▶ National : loi et code (Inria Paris), analyse statique (ENS Paris, IRISA, CEA Saclay, Inria Lille)
- ▶ International : Microsoft Research, Londres (Imperial, Facebook), Uppsala, Daejeon

# Projet d'enseignement

---

## Récapitulatif des enseignements effectués

Enseignements à Sorbonne Université, auprès d'étudiant.e.s en informatique.

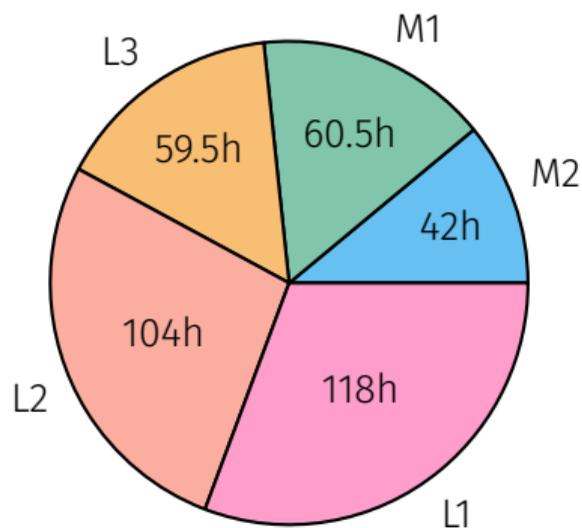
- ▶ 2018–2021 Doctorant CME (192h),
- ▶ 2021–2022 ATER temps plein (192h)

# Récapitulatif des enseignements effectués

Enseignements à Sorbonne Université, auprès d'étudiant.e.s en informatique.

- ▶ 2018–2021 Doctorant CME (192h),
- ▶ 2021–2022 ATER temps plein (192h)

Enseignement par niveau

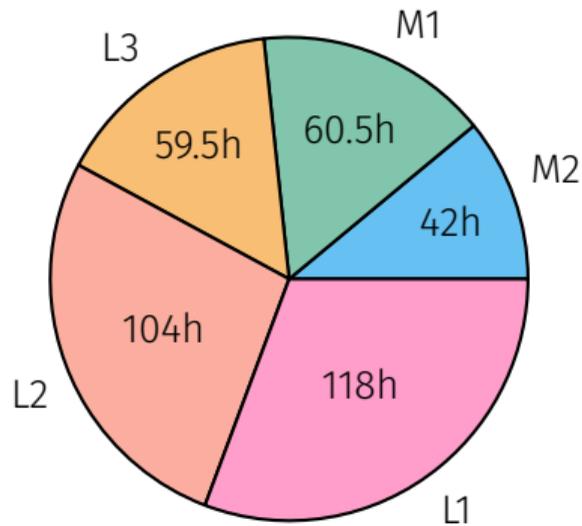


# Récapitulatif des enseignements effectués

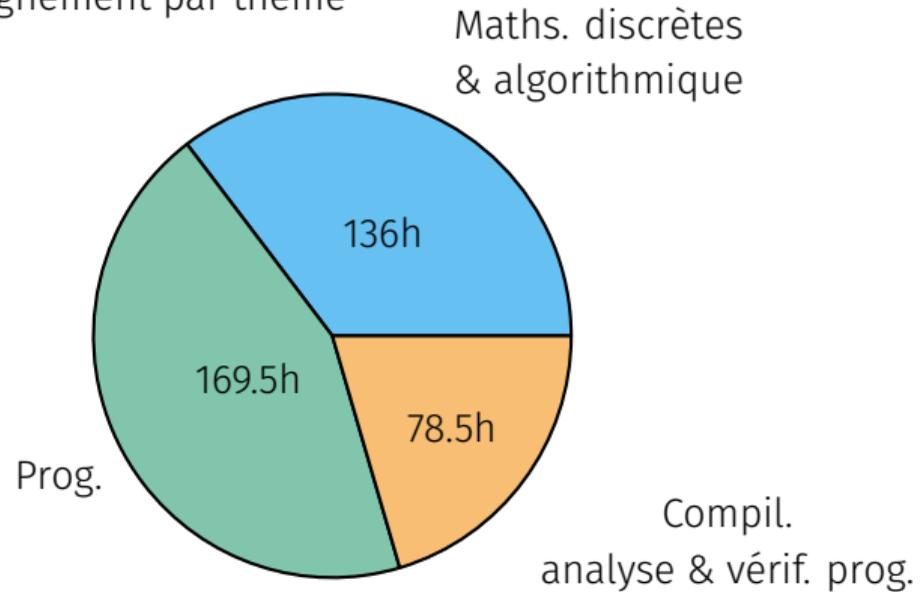
Enseignements à Sorbonne Université, auprès d'étudiant.e-s en informatique.

- ▶ 2018–2021 Doctorant CME (192h),
- ▶ 2021–2022 ATER temps plein (192h)

Enseignement par niveau



Enseignement par thème



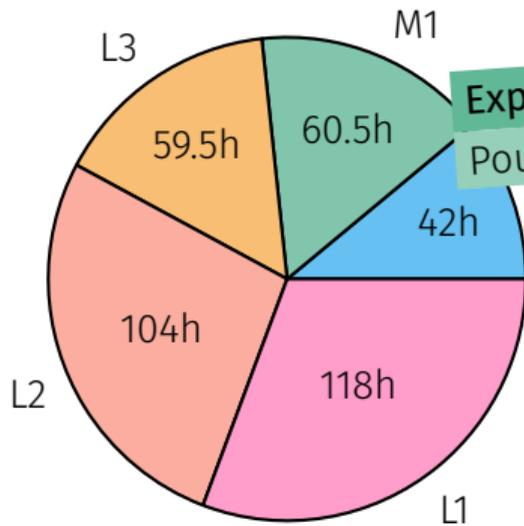
# Récapitulatif des enseignements effectués

Enseignements à Sorbonne Université, auprès d'étudiant.e.s en informatique.

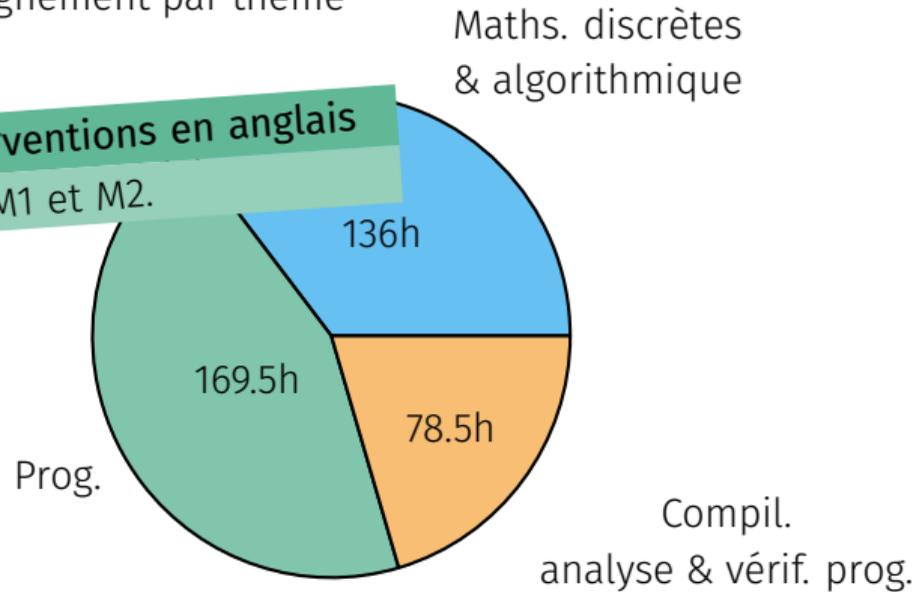
- ▶ 2018–2021 Doctorant CME (192h),
- ▶ 2021–2022 ATER temps plein (192h)

Enseignement par niveau

Enseignement par thème



**Expériences d'interventions en anglais**  
Pour les cours de M1 et M2.



- L3 – Programmation
- L3 – Projet fonctionnel
- M1 – Compilation et analyses de programmes
- M2 – Proposition de cours “Sécurité du logiciel”
- ▣ L3 – Enseignement en programmation sportive
- ▣ M1 – Projet intégré
- ▣ Prépa agreg. – TP
- L3 – Systèmes, réseaux, architecture

### Programmation (L3S1)

Actuellement : système de fiches (avec contraintes).

**Proposition** : ajout de TD & évaluation papier sur concepts clés (gestion mémoire en C).  
Expérience d'enseignement du C (TD+TP).

### Programmation (L3S1)

Actuellement : système de fiches (avec contraintes).

**Proposition** : ajout de TD & évaluation papier sur concepts clés (gestion mémoire en C).  
Expérience d'enseignement du C (TD+TP).

### UEs de systèmes, réseaux et architecture (L3)

Volontaire, d'abord pour des TD/TP puis CM à terme.

Architecture : jeu d'instruction commun avec compilation ?

### Programmation (L3S1)

Actuellement : système de fiches (avec contraintes).

**Proposition** : ajout de TD & évaluation papier sur concepts clés (gestion mémoire en C).  
Expérience d'enseignement du C (TD+TP).

### UEs de systèmes, réseaux et architecture (L3)

Volontaire, d'abord pour des TD/TP puis CM à terme.

Architecture : jeu d'instruction commun avec compilation ?

### Enseignement en programmation sportive (L3S2)

Participation au SWERC en 2016 et 2017, médaille de bronze en équipe en 2017.

## Programmation (L3S1)

Actuellement : système de fiches (avec contraintes).

**Proposition** : ajout de TD & évaluation papier sur concepts clés (gestion mémoire en C).  
Expérience d'enseignement du C (TD+TP).

## UEs de systèmes, réseaux et architecture (L3)

Volontaire, d'abord pour des TD/TP puis CM à terme.

Architecture : jeu d'instruction commun avec compilation ?

## Enseignement en programmation sportive (L3S2)

Participation au SWERC en 2016 et 2017, médaille de bronze en équipe en 2017.

## Projet fonctionnel (L3S2)

Participation aux TPs. Liens avec le cours de compilation en M1.

## Projet intégré (M1)

**Proposition :** prise en charge des enseignements en génie logiciel

- ▶ Versionnage collaboratif et évaluation de code par les pairs
- ▶ Mise en place d'une intégration continue
- ▶ Reproductibilité des résultats via des artefacts logiciels
- ▶ Évaluation de code à mi-UE

## Projet intégré (M1)

**Proposition :** prise en charge des enseignements en génie logiciel

- ▶ Versionnage collaboratif et évaluation de code par les pairs
- ▶ Mise en place d'une intégration continue
- ▶ Reproductibilité des résultats via des artefacts logiciels
- ▶ Évaluation de code à mi-UE

## Compilation et analyses de programmes (M1S1)

Actuellement : assuré par des chercheurs Inria/CNRS de l'équipe CASH.

**Proposition :** TPs à niveaux différenciés (comme en projet fonctionnel).

### Préparation à l'agrégation (M2)

- ▶ Formation à l'épreuve de TP
  - Expérience : 3h d'oraux blanc à SU sur le sujet 0
  - **Proposition** : préparation aux aspects pédagogiques liés au code
- ▶ Prise en charge des leçons sur le thème des langages de programmation
- ▶ Coordination AGPR recruté, Simon Iosti

### Préparation à l'agrégation (M2)

- ▶ Formation à l'épreuve de TP
  - Expérience : 3h d'oraux blanc à SU sur le sujet 0
  - **Proposition** : préparation aux aspects pédagogiques liés au code
- ▶ Prise en charge des leçons sur le thème des langages de programmation
- ▶ Coordination AGPR recruté, Simon Iosti

### Proposition d'UE "Sécurité du logiciel" (M1S2 ou M2)

- ▶ Approche similaire à mes recherches
- ▶ Propriétés : accès mémoire corrects, confidentialité de données, temps d'exécution
- ▶ Développement d'outils : analyseur de programmes, fuzzers
- ▶ Cibles : programmes génériques, mais aussi compilateurs

## Laboratoire

Membre élu du conseil du laboratoire du LIP6 (depuis avril 2021)

Membre du conseil des doctorants du LIP6

## National & international

Membre du collège “logiciels” dans le comité pour la science ouverte (03/2022)

Participation à la vie de la communauté

▶ PC SAS'22

▶ ERC OOPSLA'22

▶ 7 AEC entre 2019 et 2022

## Laboratoire

Membre élu du conseil du laboratoire du LIP6 (depuis avril 2021)

Membre du conseil des doctorants du LIP6

## National & international

Membre du collège “logiciels” dans le comité pour la science ouverte (03/2022)

Participation à la vie de la communauté

▶ PC SAS'22

▶ ERC OOPSLA'22

▶ 7 AEC entre 2019 et 2022

⇒ Heureux de participer au fonctionnement du DI (ainsi que du LIP) !

## Suivi du devenir des étudiant·e·s

Débouchés hors ESR ? Durée des thèses et postes ? Formations utiles ?

Suivi individuel, à M2 + 1 an, M2 + 4 ans, M2 + 7 ans.

# Candidature MCF ENS de Lyon

## Raphaël Monat – intégration dans l'équipe CASH

### Activités antérieures

- ▶ Analyse statique Python/C (LIP6)
- ▶ Calcul de l'impôt sur le revenu (Inria, MSR)

**Publications (conf. int.)** SAS'21<sup>🏆</sup>, CC'21<sup>🏆</sup>,  
ECOOP'20<sup>🏆</sup>, VSTTE'19 (invité),  
FMCAD'18, VMCAI'17.

**Publications (workshop. int.)** SOAP'20 <sup>🏆</sup>.

**Logiciels** Mopsa, 60kLoc OCaml (2018–...),  
Mlang, 10kLoc OCaml (2019–...),  
transfert à la DGFIP.

### Projet de recherche

Analyses statiques précises, sûres et efficaces  
pour les logiciels généralistes

- 1 Formalisation et analyse de sémantiques
- 2 Analyses statiques plus utilisables
- 3 Méthodes formelles pour la loi

### Projet d'enseignement

- ▶ Spécialité (prog./compil.) L3, M1, M2, agreg
- ▶ Volontaire sur archi., sys., réseau L3

Heureux de participer à la vie du DI / du LIP !