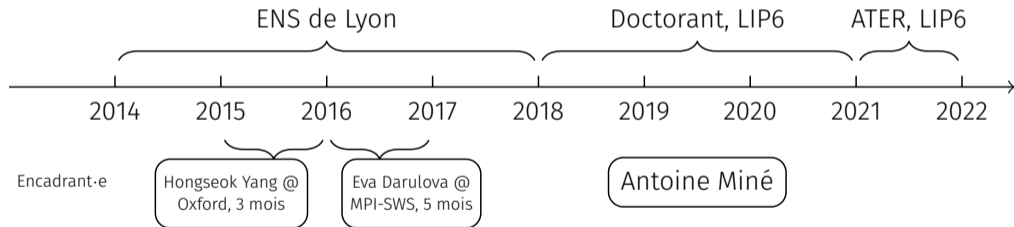


Candidature CR/ISFP Inria Lille

Raphaël Monat – intégration dans l'équipe SyCoMoRES



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

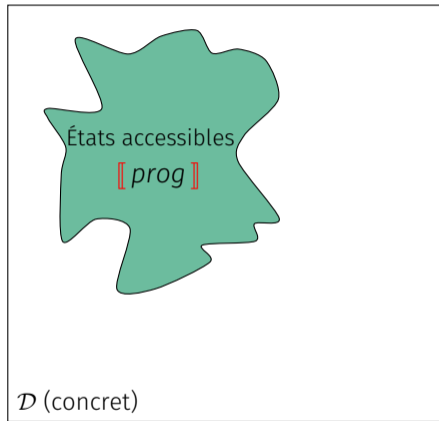
► Python, et mécanisme d'interopérabilité C (LIP6)

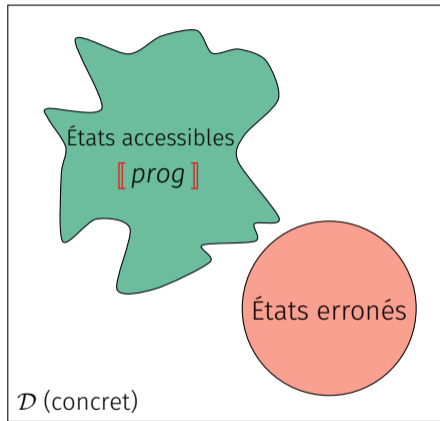
► Code de calcul de l'impôt sur le revenu (Inria, MSR)

Approche perso

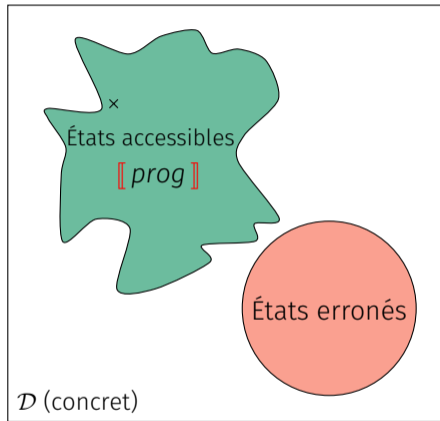
Théorie \leftrightarrow Pratique:

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

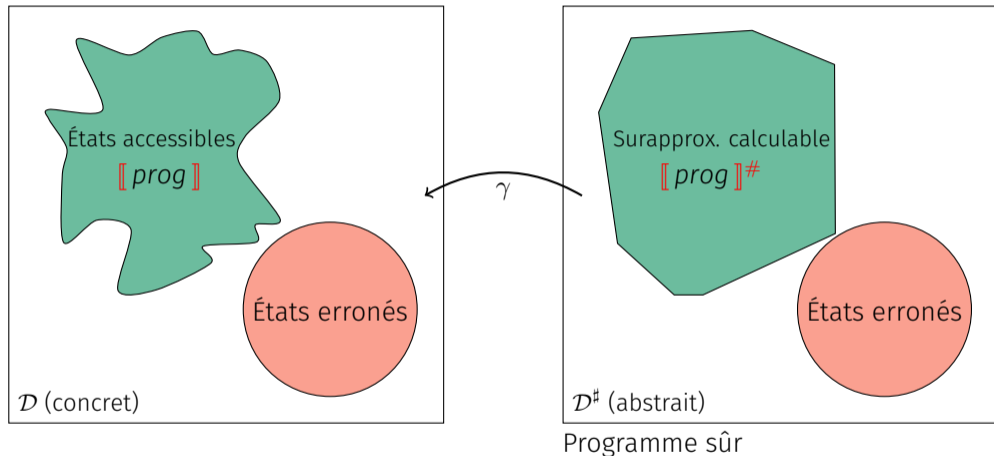




Analyse statique de programmes par interprétation abstraite

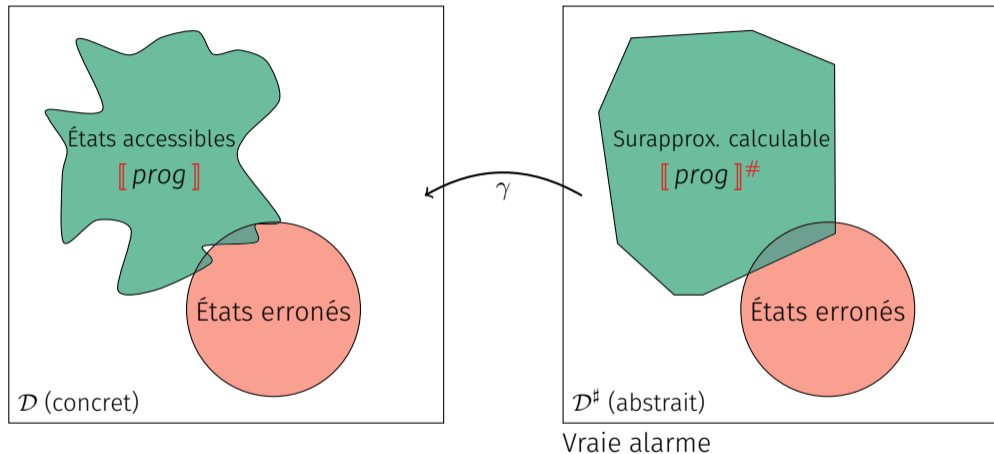


Analyse statique de programmes par interprétation abstraite



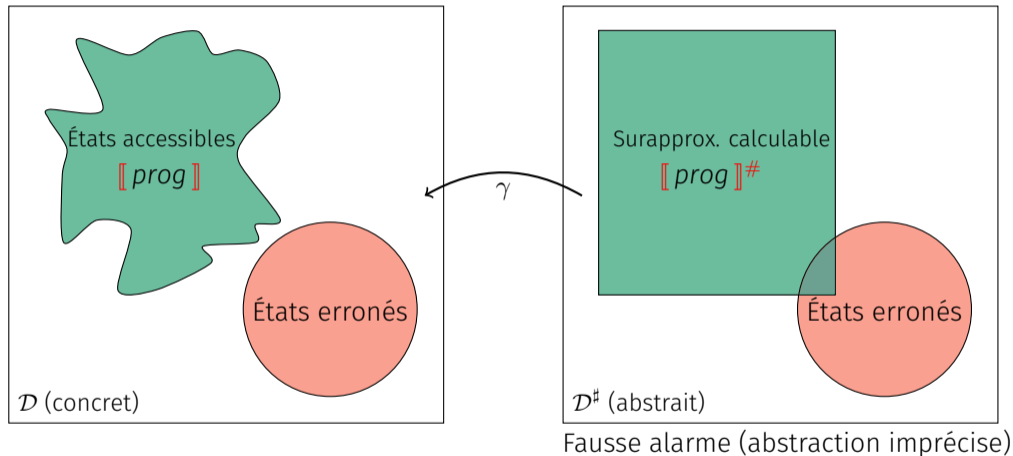
P. Cousot and R. Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". POPL 1977

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

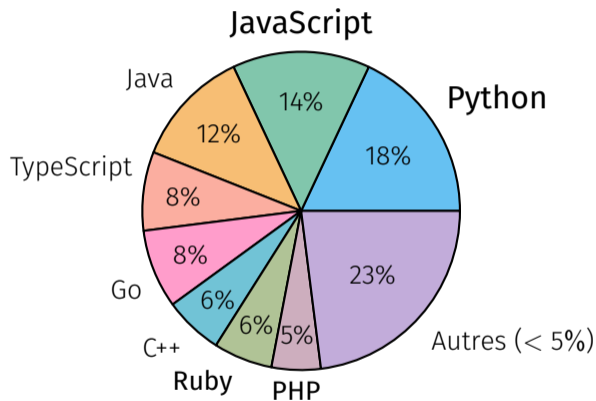
- ▶ Code généré
- ▶ Allocation dynamique

Comment démocratiser cette approche ?

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

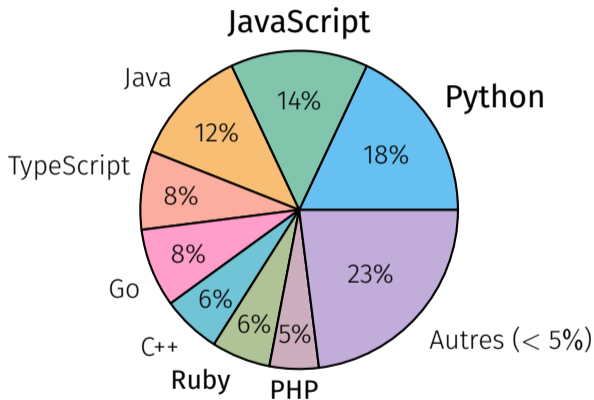
P. Cousot, R. Cousot, Feret, Mauborgne, Miné, Monniaux, and Rival. "Combination of Abstractions in the Astrée Static Analyzer". ASIAN 2006

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^{1,2} $[\cdot]_{py} \rightsquigarrow [\cdot]_{py}^\#$

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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^{1,2} `[·].py` \rightsquigarrow `[·].py#`

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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^{1,2} `[·]_py` \rightsquigarrow `[·]_py^#`

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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^{1,2} `[·]_py` \rightsquigarrow `[·]_py^#`

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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^{1,2} []_py \rightsquigarrow []_py#

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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^{1,2} $[\cdot]_{py} \rightsquigarrow [\cdot]_{py}^{\#}$

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

Analyse multilingage Python/C³

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

Implémentation dans Mopsa⁴

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

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

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

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

⁴Journault, 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 précédents⁵ : JNI  Java, peu précis

⁵Tan 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

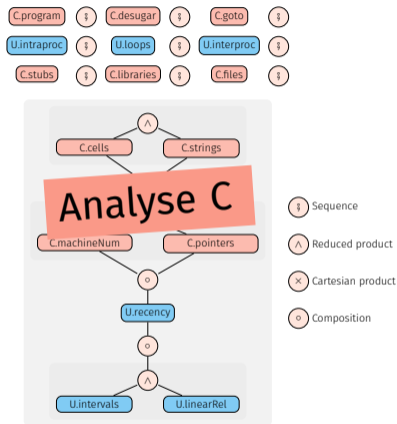
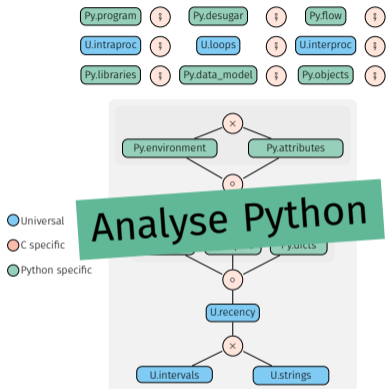
Séparation de l'état et synchronisation réduite

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

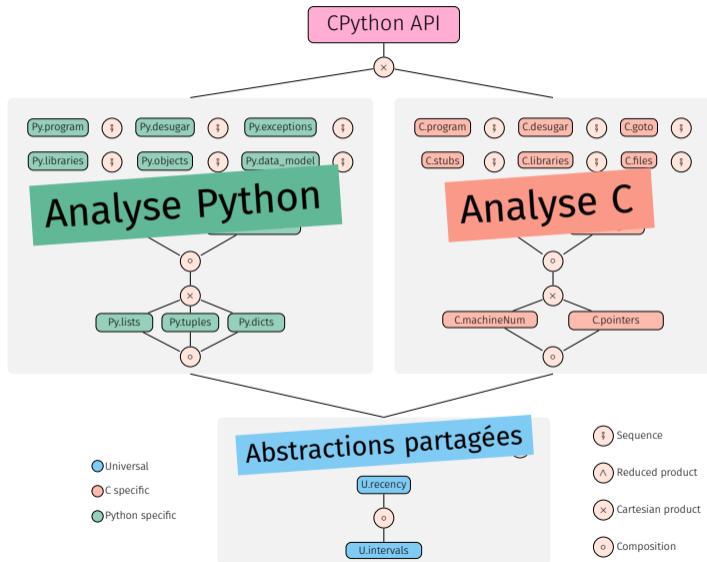
⇒ Séparation relative de l'état

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

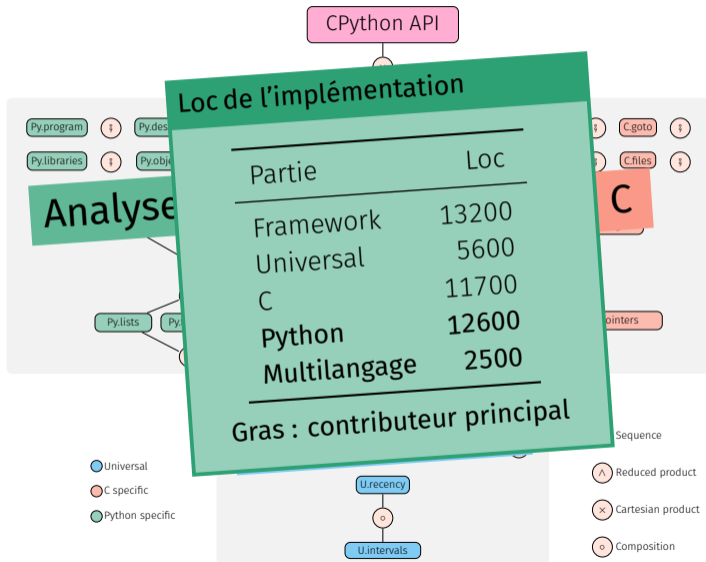
Analyse multilangage – implémentation



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)
bitarray	5841	159/216	1.6s	94.9%	(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)

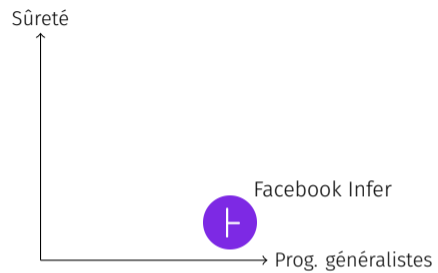
- ▶ **SAS'21** , **ECOOP'20** , VSTTE'19 (invité), **SOAP@PLDI'20** , JFLA'21 (fr, outil)
- ▶ Orateur invité au Facebook Testing and Verification Symposium 2021
- ▶ Mopsa (LGPL v3, 60kLoc OCaml), contributeur principal depuis septembre 2018

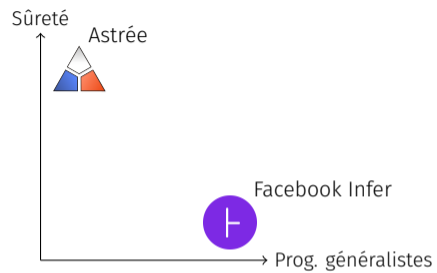
Code de calcul de l'impôt sur le revenu

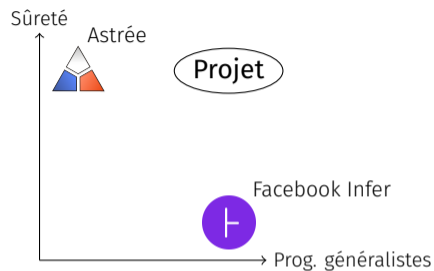
Merigoux (Prosecco, Inria Paris) et Protzenko (Microsoft Research)

- ▶ **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

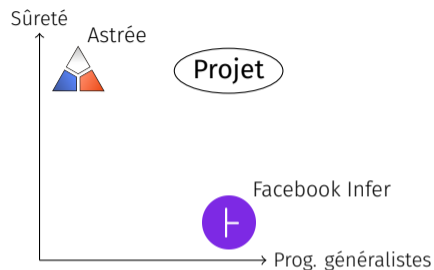
Projet de recherche et intégration







⇒ 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



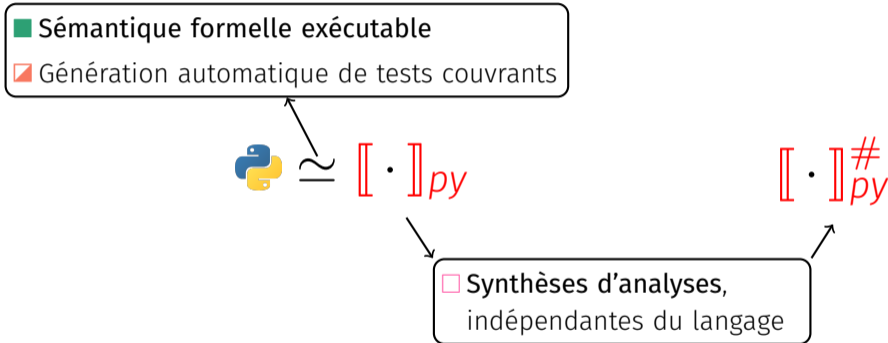
- Sémantique formelle exécutable
- ▣ Génération automatique de tests couvrants

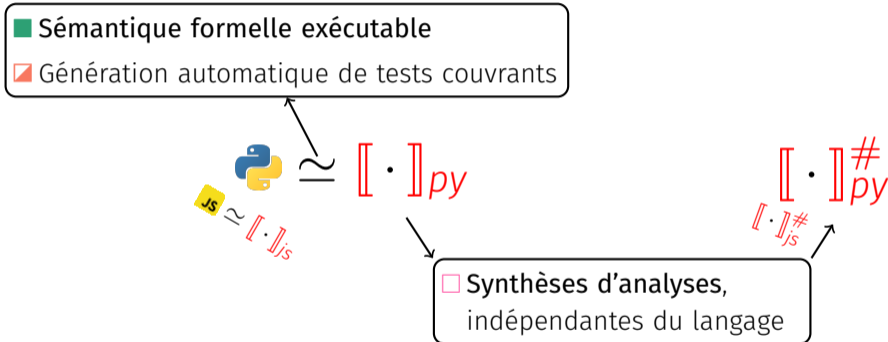


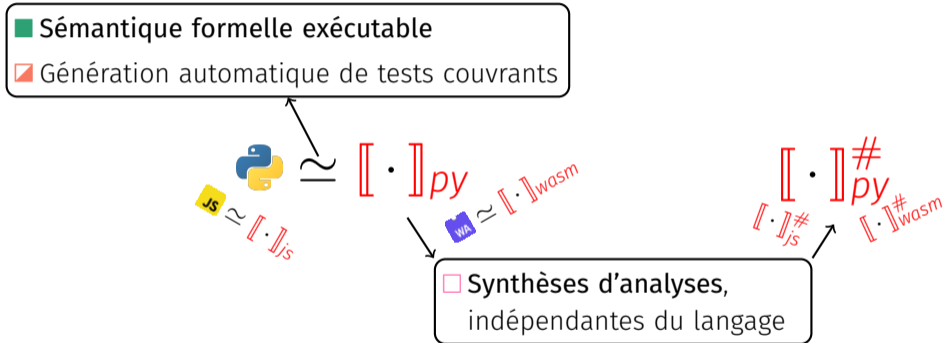
\approx

$\llbracket \cdot \rrbracket_{py}$

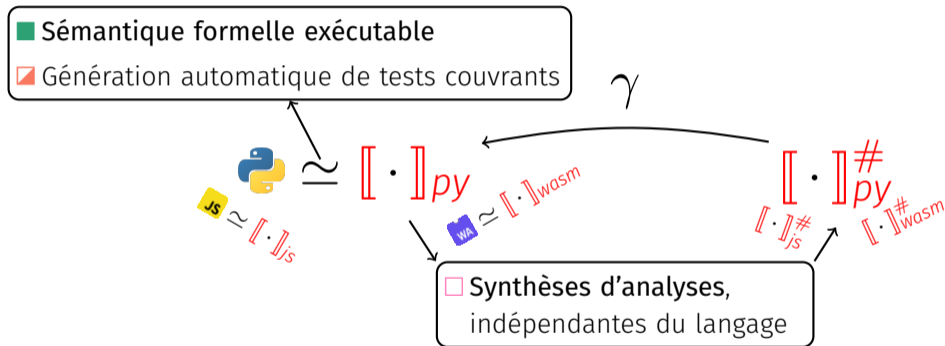
$\llbracket \cdot \rrbracket_{py}^{\#}$



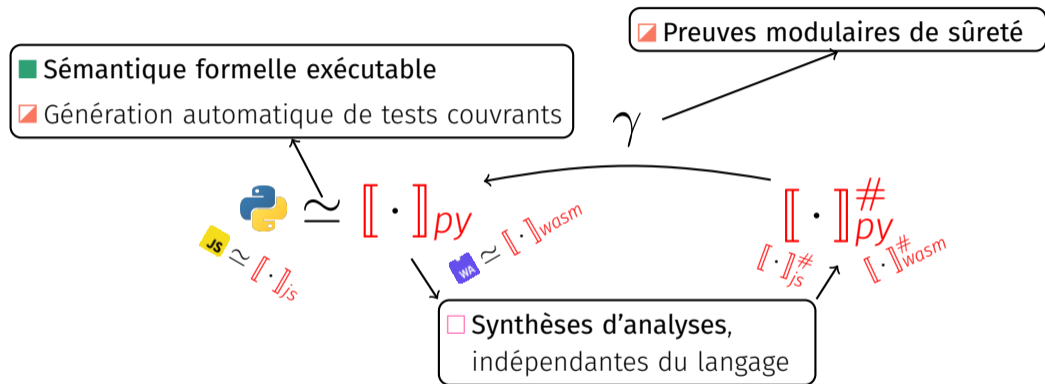




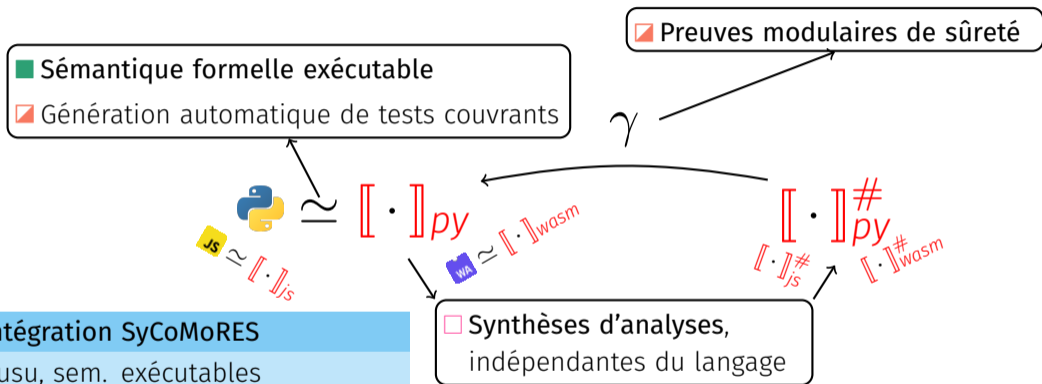
Axe I – Formalisation et analyse sûre de sémantiques complexes



Axe I – Formalisation et analyse sûre de sémantiques complexes



Axe I – Formalisation et analyse sûre de sémantiques complexes

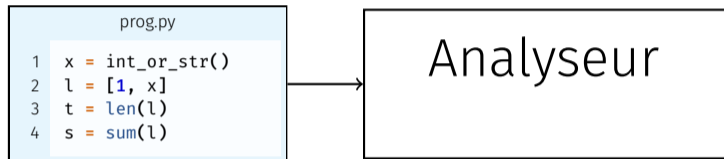


Intégration SyCoMoRES

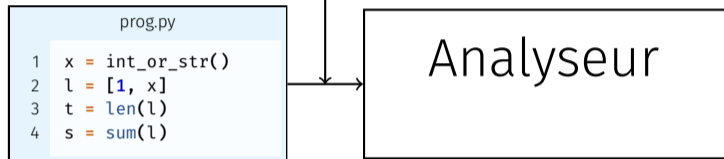
Rusu, sem. exécutables

Modularité au cœur de SyCoMoRES

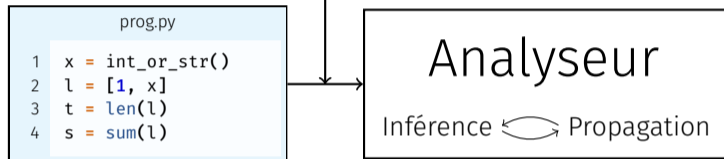
⇒ Apport expertise sem. langages dynamiques



□ Analyses compositionnelles de fonctions



□ Analyses compositionnelles de fonctions



Axe II – Rendre les analyses statiques plus utilisables

□ Analyses compositionnelles de fonctions

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



▣ Analyses incrémentales

Axe II – Rendre les analyses statiques plus utilisables

Analyses compositionnelles de fonctions

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



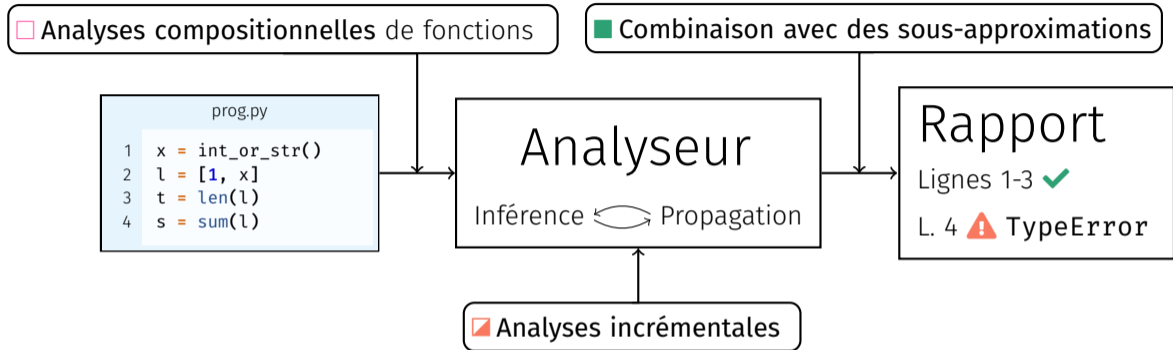
Rapport

Lignes 1-3 ✓

L. 4 ⚠ TypeError

Analyses incrémentales

Axe II – Rendre les analyses statiques plus utilisables



Axe II – Rendre les analyses statiques plus utilisables

□ Analyses compositionnelles de fonctions

■ Combinaison avec des sous-approximations

```
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

Intégration SyCoMoRES

Ballabriga, Forget, Lipari

Analyse de sécurité de binaire

Parallèles (typage) avec Python

⇒ Renforce branche interp. abs.

Ballabriga, Forget, Gonnord, Lipari, and Ruiz. "Static Analysis of Binary Code with Memory Indirections Using Polyhedra". VMCAI 2019

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_deuxieme_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

Axe III – Méthodes formelles pour la loi

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

- ▶ Pas de correspondance
- ▶ 2019 ~> 2020

Catala, un nouveau langage

Article D52

I - Pour l'application de l'art...
majoration pour âge prévue...
1° Lorsque le ménage ou la...
égal au plafond défini au I d...
familiales sont fixés, en pour...
a) 32 % pour le deuxième enf...

```
*** catala
champ d'application AL
définition montant_i
ressources_ménage :
conséquence égal à
si nombre de enfant
alors prestations_familiales.base_mensuelle * € 32 %
sinon 0 €
***
```

- Développement de Catala
- ☑ Vérification de la loi
- ☑ Simulation parallèle efficace
- ☐ Catala dans les administrations

Apport d'un nouveau domaine prometteur et collaborations avec DGFIP & OCamlPro.

Collaborations avec équipe Prosecco (Inria Paris)

Légende ■ court terme ☑ moyen terme ☐ long terme

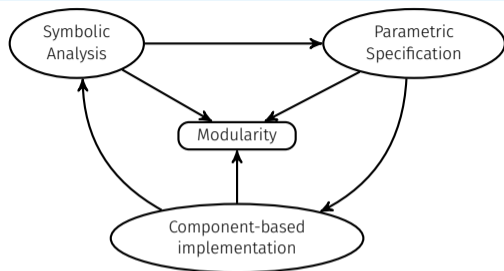
en littéraire, par /développeur·euse
faut, pour la loi
depuis janvier 2022
en dates/durées
ique de Catala

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 SyCoMoRES



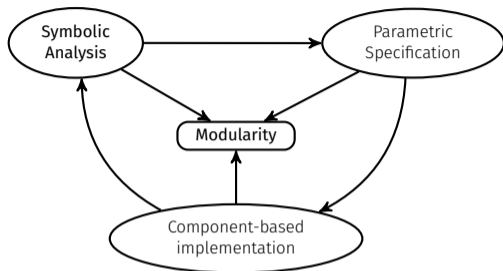
Ontologie des termes de l'équipe SyCoMoRES

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 SyCoMoRES



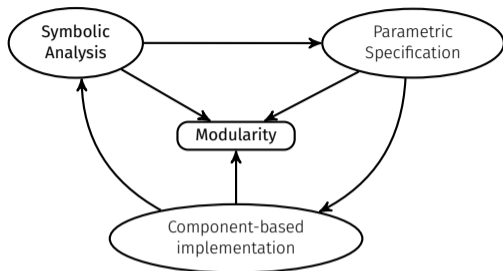
Ontologie des termes de l'équipe SyCoMoRES

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 SyCoMoRES



Ontologie des termes de l'équipe SyCoMoRES

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 inter-centre Inria, international

- ▶ Inria :
 - écosystème OCaml,
 - loi et code (Prosecco),
 - analyse statique (Antique, CASH, Celtique)
- ▶ International : Microsoft Research, Londres (Imperial, Facebook), Uppsala, Daejeon

Candidature CR/ISFP Inria Lille

Raphaël Monat – intégration dans l'équipe SyCoMoRES

Activités antérieures

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

Publications (conf. intl.) SAS'21[🏆], CC'21[🏆],
ECOOP'20[🏆], VSTTE'19 (invité),
FMCAD'18, VMCAI'17.

Publications (workshop. intl.) SOAP'20 [🏆].

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

Responsabilités collectives

- ▶ Membre élu du conseil du laboratoire LIP6
- ▶ Membre du collège "logiciels" du CoSO
- ▶ PC SAS'22, ERC OOPSLA'22, 7 AEC (19-22)

⇒ Heureux de participer à la vie du centre !

Table des annexes

- | | | | |
|---|-----------------------------------|----|---------------------------------------|
| 1 | Mise à jour du dossier | 10 | Analysis of the multilanguage example |
| 2 | Rice's impossibility theorem | 11 | A formal semantics for M |
| 3 | Concrete semantics – example | 12 | DGFIP's legacy architecture |
| 4 | Dual dynamic typing in Python | 13 | Mlang's architecture |
| 5 | Comparison of the Python analyses | 14 | Mlang's correctness |
| 6 | Selectivity of the value analysis | 15 | Code optimization |
| 7 | Example of multilanguage code | 16 | Contributions around the tax code |
| 8 | Example of multilanguage state | | |

- ▶ Co-encadrement du stage M2 de Milla Valnet (Mars – Juillet 2022)
(stagiaire du MPRI, avec Antoine Miné, réunions hebdomadaires de 2h).
- ▶ Membre du comité de programme de SAS'22 (conf. spécialisée int. abs.).
- ▶ Membre du comité de programme externe de OOPSLA'22 (conf. généraliste m.f.).
- ▶ Membre du collège logiciels du comité pour la science ouverte
(dirigé par R. Di Cosmo et F. Pellegrini).