

Audition pour un poste de professeur assistant Monge

Ambroise Lafont

16 mai 2023

Postdocs : University of Cambridge (2022-...)
University of New South Wales (2020-2022)

Thèse : LS2N, Nantes (2016-2019)
Master : MPRI, Ecole Polytechnique

Plan

1. Recherche

- Travaux précédents et en cours
- Projet

2. Enseignement

- Expérience
- Projet

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
- Sémantique opérationnelle

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- **Syntaxe**

- Substitution
- Unification

Thèse, CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

- Sémantique opérationnelle

Assistants de
preuve

- Fondements
- Ergonomie

- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - **Substitution**¹
 - Unification
- Sémantique opérationnelle

Thèse, CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

¹ avec B. Ahrens, A. Hirschowitz, T. Hirschowitz, M. Maggesi

Des disciplines pour spécifier des syntaxes avec substitution

Exemple du λ -calcul différentiel

- Définition
- Propriétés de substitution

10 pages

[Ehrhard-Regnier '03]

Découpage en

LMCS 2022

- Une partie réutilisable (théorie des arités / équations)
- Une partie spécifique de **quelques lignes**
(3 arités, une équation)

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - **Substitution**
 - Unification
- Sémantique opérationnelle

Thèse, CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution Thèse, CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - **Unification** (postdoctorat actuel¹) Preprint 2022
- Sémantique opérationnelle

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

¹ avec N. Krishnaswami

Unification

- Programmation logique (Prolog)
- Inférence de types (ocaml, haskell, ...)
- **Assistants de preuve**



Je veux prouver $a + 0 = a$ avec le schéma d'induction

$$\frac{P(0) \quad P(i) \Rightarrow P(i + 1)}{P(n)}$$

Unification (ordre supérieur)
Instancier les **métavariab**les P et n
pour que $P(n) = (a + 0 = a)$



- Qui est P ?
 $x \mapsto x + 0 = x$
- Qui est n ?
 a



Indécidable



Fragment décidable identifié par Miller (1991) pour le λ -calcul

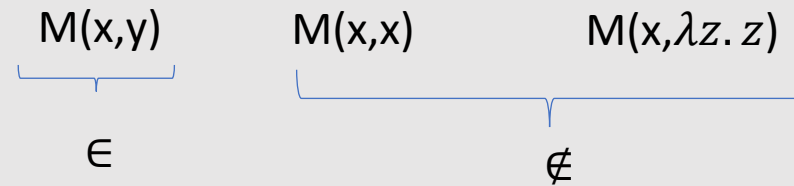
Unification à la Miller (pattern)

[Miller 1991]

- Fragment décidable de l'unification d'ordre supérieur pour le λ -calcul simplement typé (modulo $\beta\eta$)

Caractérisation du fragment :

Arguments des métavariabes = variables distinctes



Ma feuille de route (Neel Krishnaswami, janvier 2022)

- Un « fragment de Miller » envisagé pour toute *signature liante*
- Un algorithme envisagé
- La question : preuve de correction ?

Ce que j'ai fait pendant mon postdoctorat

- Début de mécanisation en Coq (bug dans l'algo de Neel !)
- Reformulation : *unificateur le plus général* \cong coégalisateur
(dans une catégorie dépendant de la signature).
- 1^{ère} description de cette catégorie
 - « Fragment de Miller » d'une catégorie à la Fiore & al.
 - Preuve complète correspondante (sur papier)
- 2^{ème} description plus simple (et originale) de cette catégorie
⇒ Preuve complète plus simple
- Découverte que cette preuve se généralise naturellement à une classe bien plus large de syntaxes incluant le *système F* par exemple.

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution
 - **Unification**
- Sémantique opérationnelle

CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- **Sémantique opérationnelle**
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - **Substitution**¹ Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022


Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

¹ avec B. Ahrens, A. Hirschowitz, T. Hirschowitz, M. Maggesi

Notions formelles de langages de programmation

Réécriture d'ordre supérieur

 Congruence imposée

Formats (GSOS, ...)

 Premier ordre

... (variantes catégoriques)

Une contribution issue de ma thèse

Monades de réduction
(POPL 2020)

Notion de spécification
Propriétés de substitution automatiques

Exemples :

- *λ -calcul avec β -réduction faible*
- *λ -calcul avec substitution explicite*
[Kesner '09]
- ...

$$\frac{t \rightarrow u}{t[\sigma] \rightarrow u[\sigma]}$$

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - **Substitution** Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - **Equivalences de programmes¹** LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie

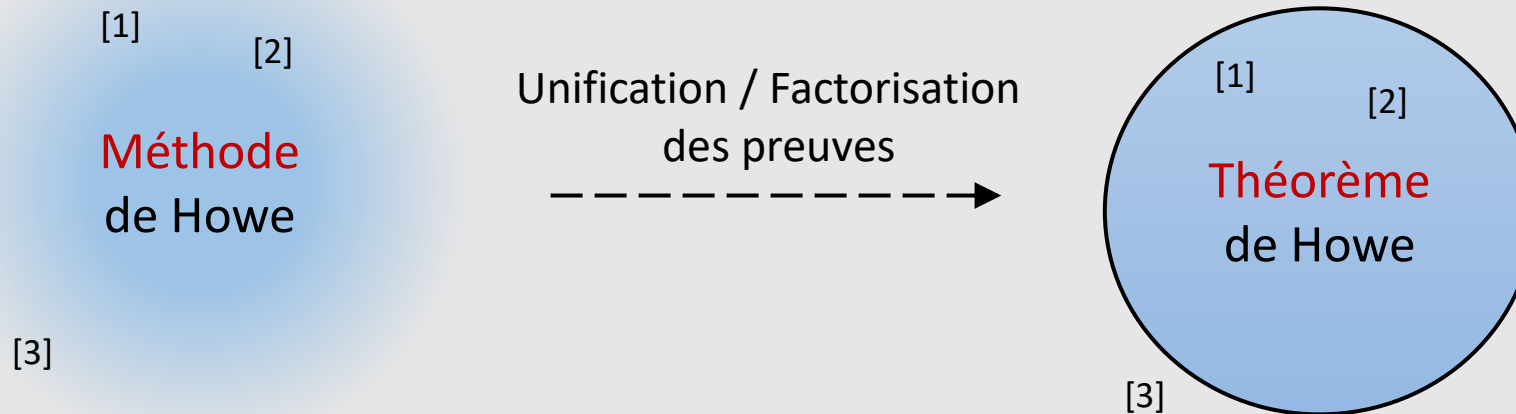
- Compilation certifiée

¹ avec P. Borthelle, T. Hirschowitz

Congruence de la *bisimilarité*

Une preuve séparée / différente
par langage

Un théorème pour une classe de
langages



Axe de recherche en cours
(**LICS** 2020, **LMCS** 2022, preprint 2023)

Exemples

- [1] Gordon, “Bisimilarity as a theory of functional programming”, 1999
- [2] Biernacki-Lenglet, “Applicative Bisimulations for Delimited-Control Operators”, 2012
- [3] Lenglet-Schmitt, “Howe’s Method for Contextual Semantics”, 2015

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - **Equivalences de programmes** LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution
 - Unification
- Sémantique opérationnelle
 - Substitution
 - Equivalences de programmes

CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Thèse, POPL 2020, FSCD 2020, LMCS 2022
LICS 2020, LMCS 2022

Assistants de
preuve

- **Fondements¹**
- Ergonomie
- Compilation certifiée

TYPES 2019

¹ avec A. Kaposi, A. Kovács

La théorie des types comme sa propre métathéorie

Sémantique Types interprétés par des ω -groupoïdes (sémantique homotopique)

Formalisation : tout type est un ω -groupoïde (définition et preuve)



Techniques similaires

Syntaxe

Construction d'une classe de types inductifs avancés permettant de définir la théorie des types en théorie des types



TYPES 2019

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- **Fondements** TYPES 2019
- Ergonomie
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

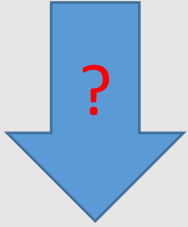
- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements TYPES 2019
- **Ergonomie**
- Compilation certifiée

Mécanisation de diagrammes

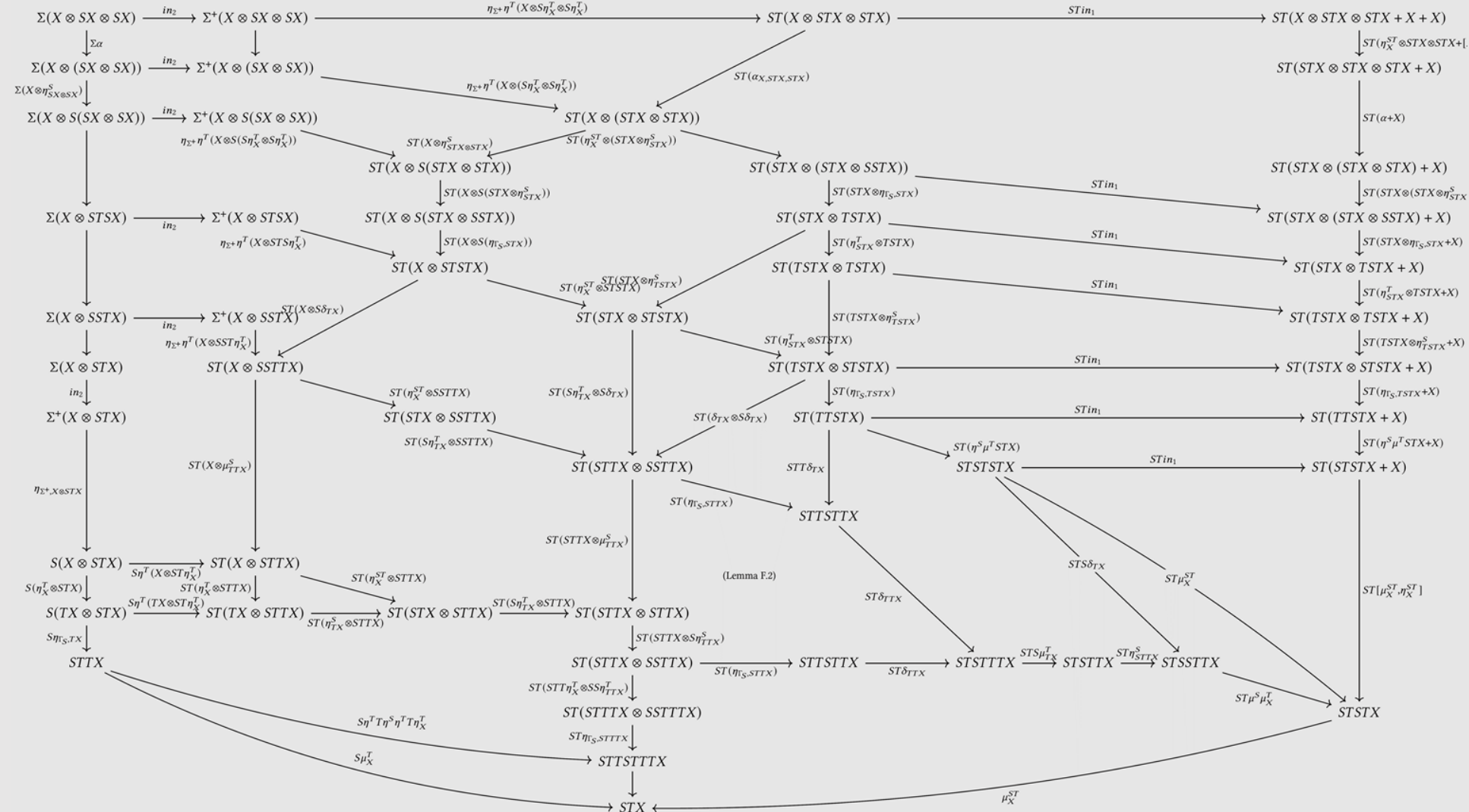
une preuve →
(travail en cours sur Howe)



PREUVE MÉCANISÉE



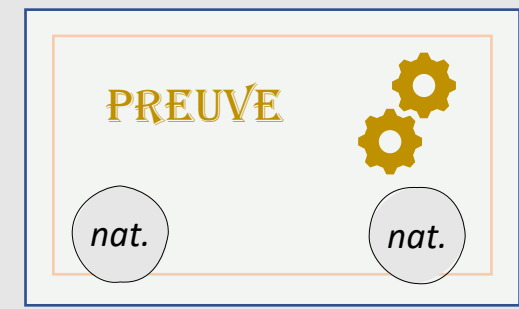
texte





Enoncé

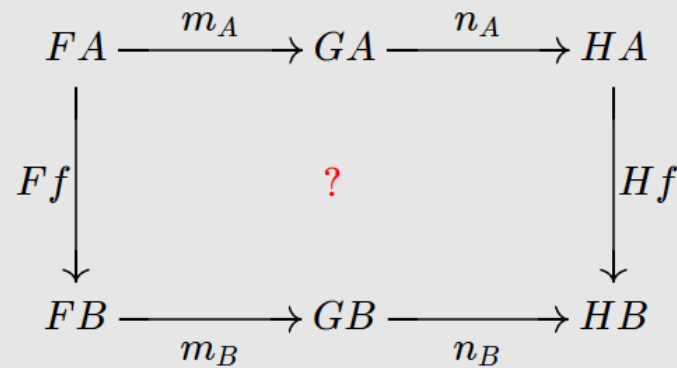
$$m_A \circ n_A \circ Hf = Ff \circ m_B \circ n_B$$



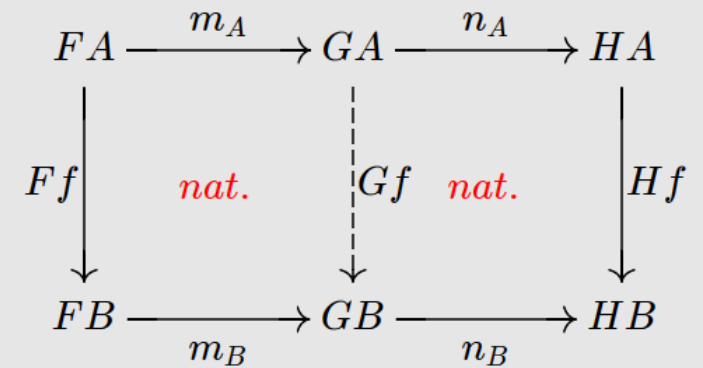
Affiche

Génère

Éditeur de diagrammes
(application Web¹)



Actions de
l'utilisateur



Projet ANR CoREACT (2023-2027)

porté par Nicolas Behr, IRIF

*Méthodologie pour le raisonnement diagrammatique en Coq,
s'appuyant sur mon éditeur.*

¹ Accessible depuis ma page web.

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements TYPES 2019
- **Ergonomie** Application web
- Compilation certifiée

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
 - Unification Preprint 2022
- Sémantique opérationnelle
 - Substitution Thèse, POPL 2020, FSCD 2020, LMCS 2022
 - Equivalences de programmes LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements TYPES 2019
- Ergonomie Application web

- **Compilation certifiée**



- Conversion des clôtures (stage avec X. Leroy)
- Cogent (1^{er} postdoctorat)

Mémoire de Master

POPL 2023 avec l'équipe Cogent

Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution
 - Unification
- Sémantique opérationnelle
 - Substitution
 - Equivalences de programmes

CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Thèse, POPL 2020, FSCD 2020, LMCS 2022
LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée
- **Conversion des clôtures**
(stage avec X. Leroy)
- Cogent (1^{er} postdoctorat)

TYPES 2019
Application web

Mémoire de Master

POPL 2023 avec l'équipe Cogent



Travaux précédents et en cours

Théorie des
langages de
programmation

- Syntaxe
 - Substitution
 - Unification
- Sémantique opérationnelle
 - Substitution
 - Equivalences de programmes

CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

Thèse, POPL 2020, FSCD 2020, LMCS 2022
LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements
- Ergonomie
- Compilation certifiée
 - Conversion des clôtures
(stage avec X. Leroy)
 - **Cogent** (1^{er} postdoctorat)

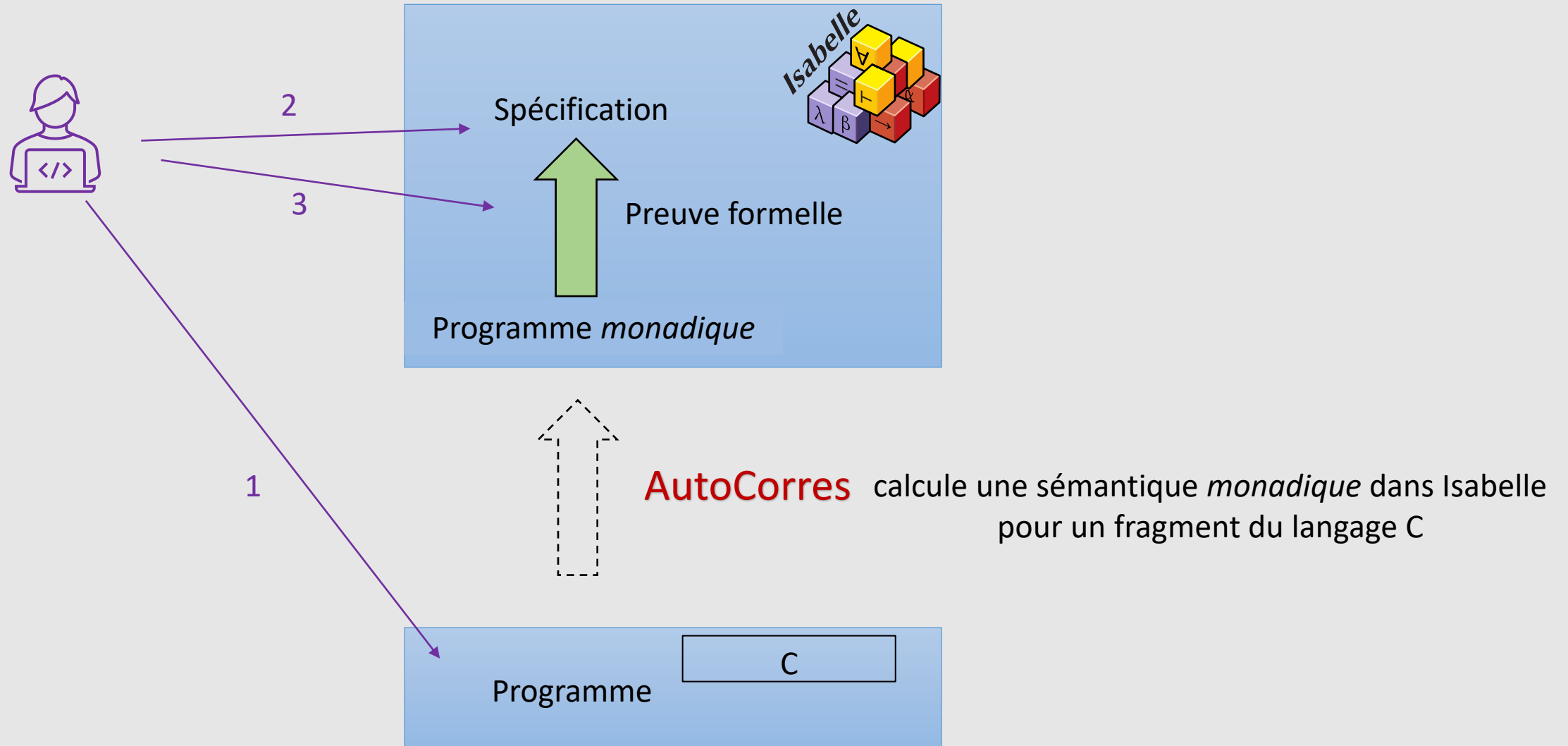


TYPES 2019
Application web

Mémoire de Master

POPL 2023 avec l'équipe Cogent

Raisonner sur du code C en Isabelle



La sémantique monadique d'AutoCorres

C

Type A

$f: A \rightarrow B$

AutoCorres

Type A_m

$f_m: A_m \rightsquigarrow B_m$

$A_m \times \text{state} \rightarrow (B_m \times \text{state}) \text{ option}$

Effets :

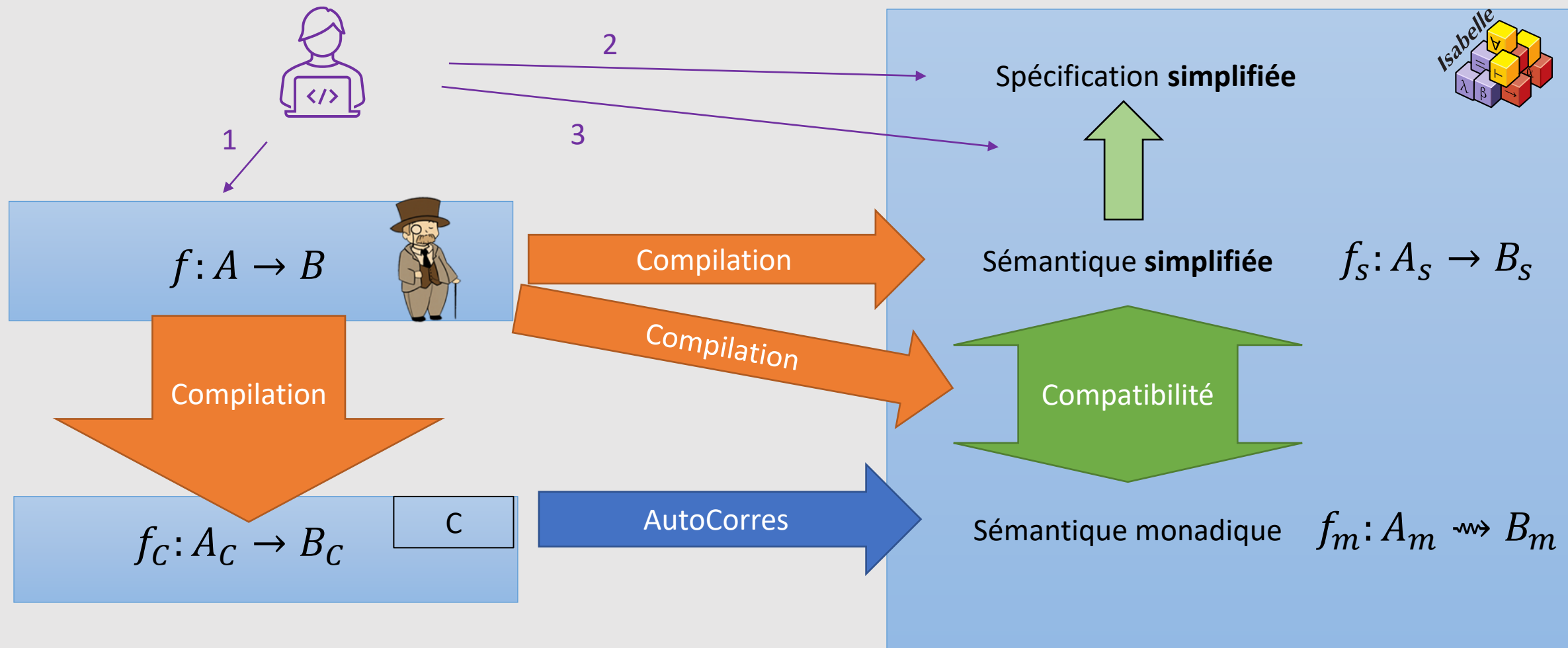
- État mémoire
- Erreur
- (non déterminisme)





Cogent : Une sémantique simplifiée

Un langage fonctionnel total pour décrire un sous-fragment **sûr** d'AutoCorres, muni d'une sémantique **simplifiée**



Comparaison des sémantiques

Sémantique monadique	Sémantique simplifiée
Type T_m	Type T_s
$f_m: A_m \rightsquigarrow B_m$	$f_s: A_s \rightarrow B_s$

Simplifications :

- Plus de pointeur / mémoire

$T_m = \text{int}^*$	$T_s = \text{int}$
----------------------	--------------------

- Sémantique fonctionnelle pure

$f_m: \text{int} \rightsquigarrow \text{int}$	$f_s: \text{int} \rightarrow \text{int}$
---	--

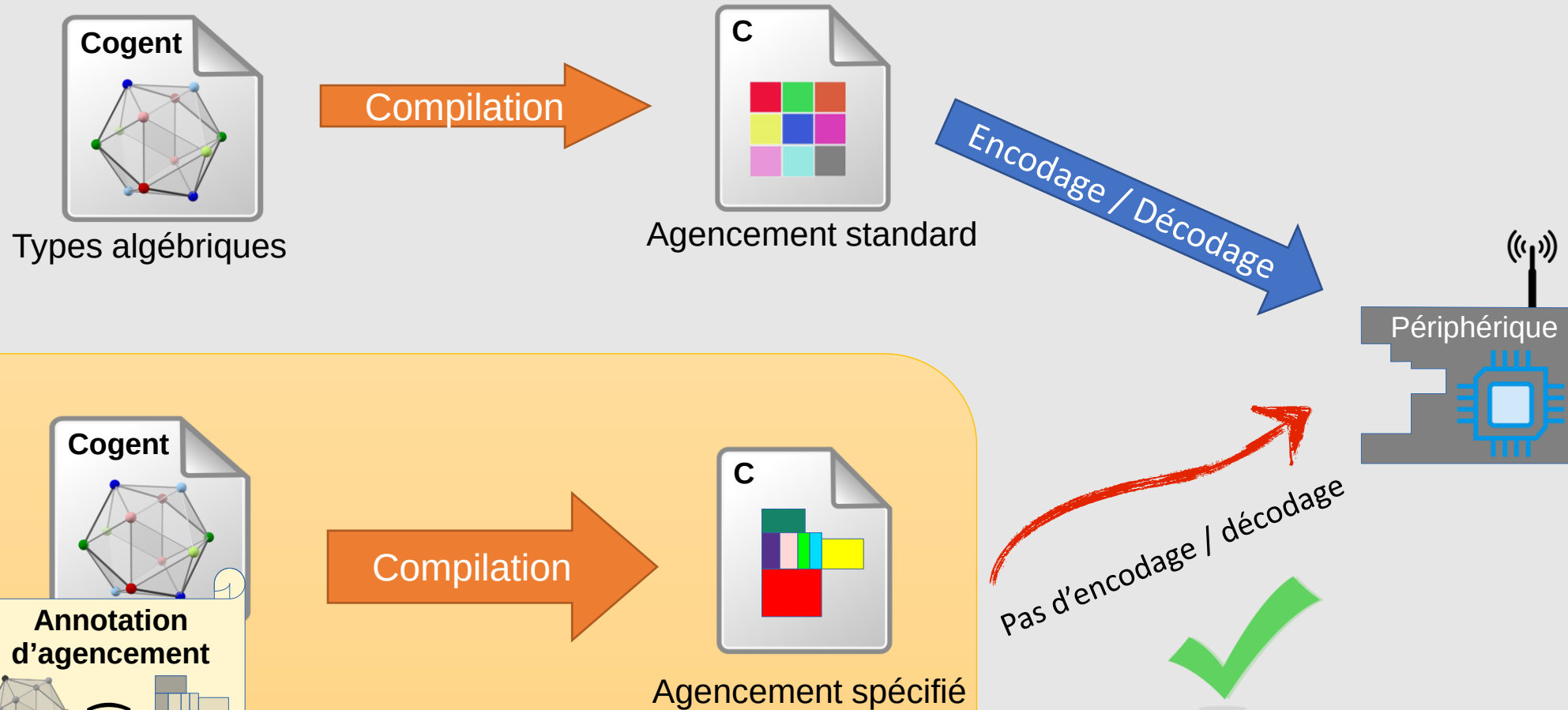
Compatibilité entre les deux sémantiques
Relation $T_r \subset \text{state} \times T_m \times T_s$
(f_m, f_s) est compatible avec (A_r, B_r)

« (f_m, f_s) envoie des valeurs reliées sur des valeurs reliées »
(en particulier, f_m n'échoue pas)

$T_r = \{(s, p, s(p))\}$

$f_m(s, n) = \text{Some}(s', f_s(n))$

Contrôler l'agencement des données



Extension Dargent

Cogent⁺ = Cogent + Dargent

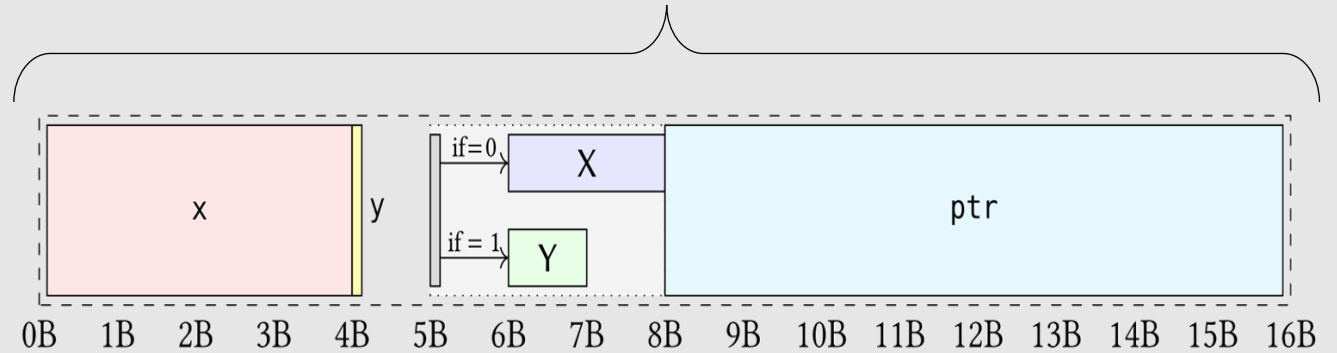
Cogent enrichi avec la possibilité d'annoter des types records par des agencements explicites.

Syntaxe dédiée

Type Cogent

```
type Example = {  
  struct : #{x : U32, y : Bool},  
  ptr : {...},  
  sum : <X U16 | Y U8>  
}
```

Agencement possible



Compilation Cogent⁺ vers C

```
type Example = {  
  struct : #{x : U32, y : Bool},  
  ptr : {...},  
  sum : <X U16 | Y U8>  
}
```

Compilation

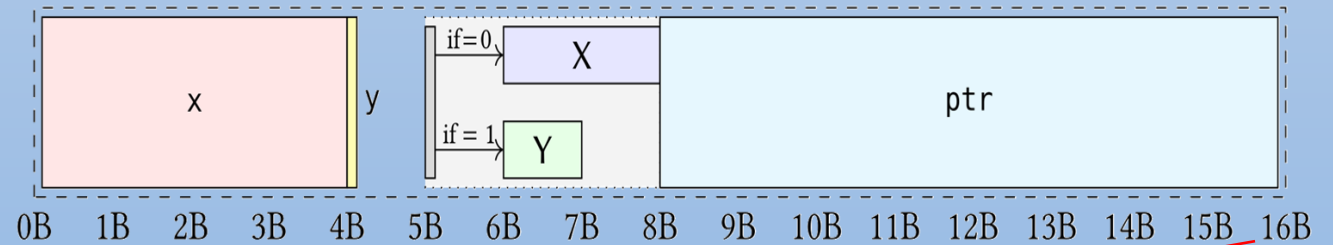
Type : tableau de 16 octets

Getters/setters pour
chaque champ

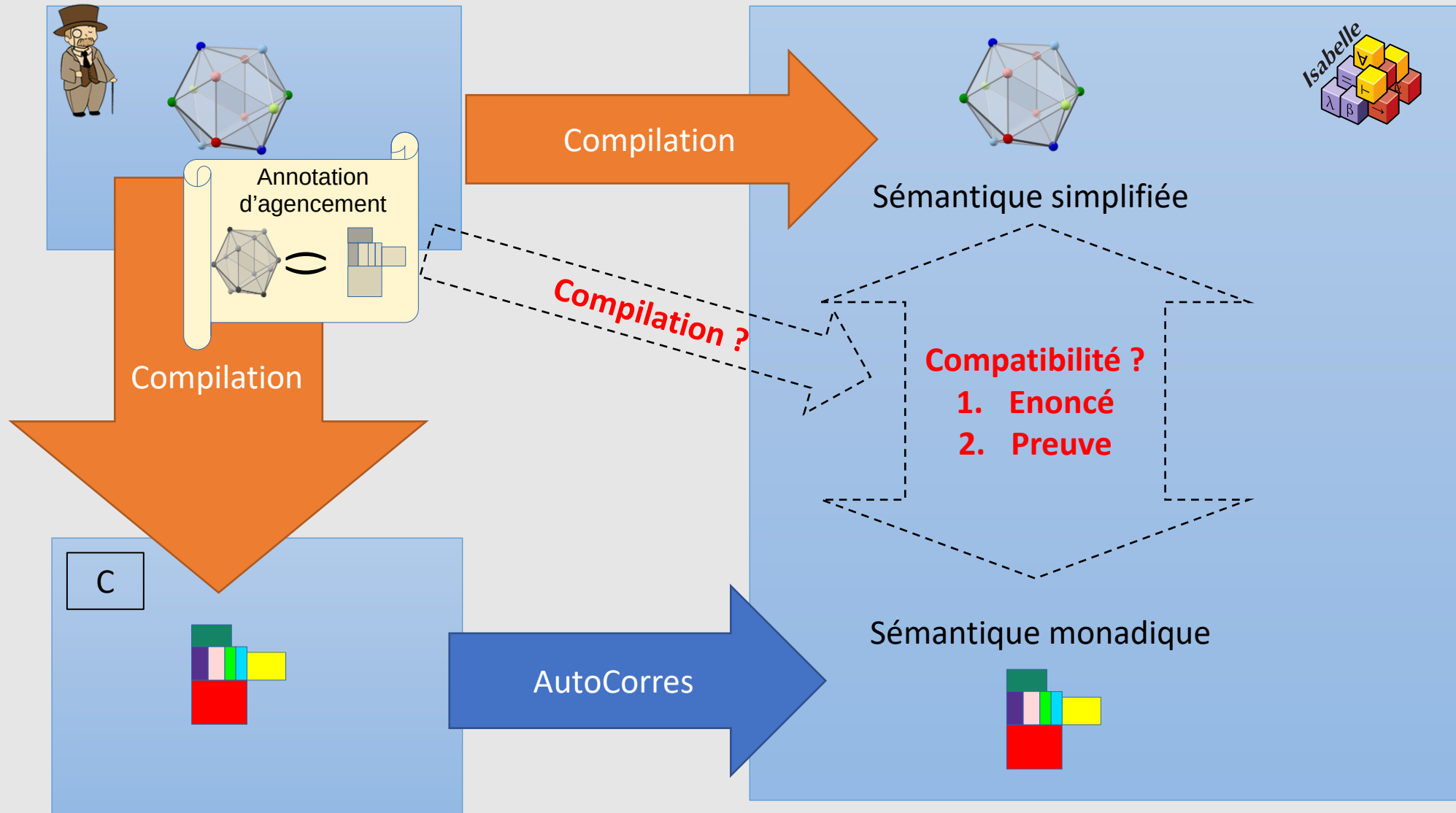
$get_{ptr} : \text{byte}[16] \rightarrow ptr$

$set_{ptr} : \text{byte}[16] \times ptr \rightarrow ()$

C



Ma mission (2020)

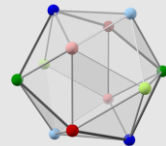
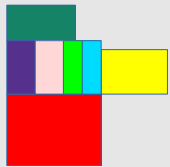


Enoncer la compatibilité

Sémantique monadique	Sémantique simplifiée
Type T_m	Type T_s
$f_m: A_m \rightarrow B_m$	$f_s: A_s \rightarrow B_s$

- Pour un type non annoté, même T_r qu'avant
- Quid d'un type annoté ?

$T_m = \text{byte}[n]$	$T_s = \{ \dots, x : A, \dots \}$
------------------------	-----------------------------------



Compatibilité entre les deux sémantiques
Relation $T_r \subset state \times T_m \times T_s$?
(f_m, f_s) est compatible avec (A_r, B_r)

Relation de décodage : « $T_r = \{ (\dots, t, decode(t)) \}$ »

Décode le tableau
selon l'agencement

$$decode : tableau \mapsto \left\{ x := get_x^{...}(tableau) \right\}$$

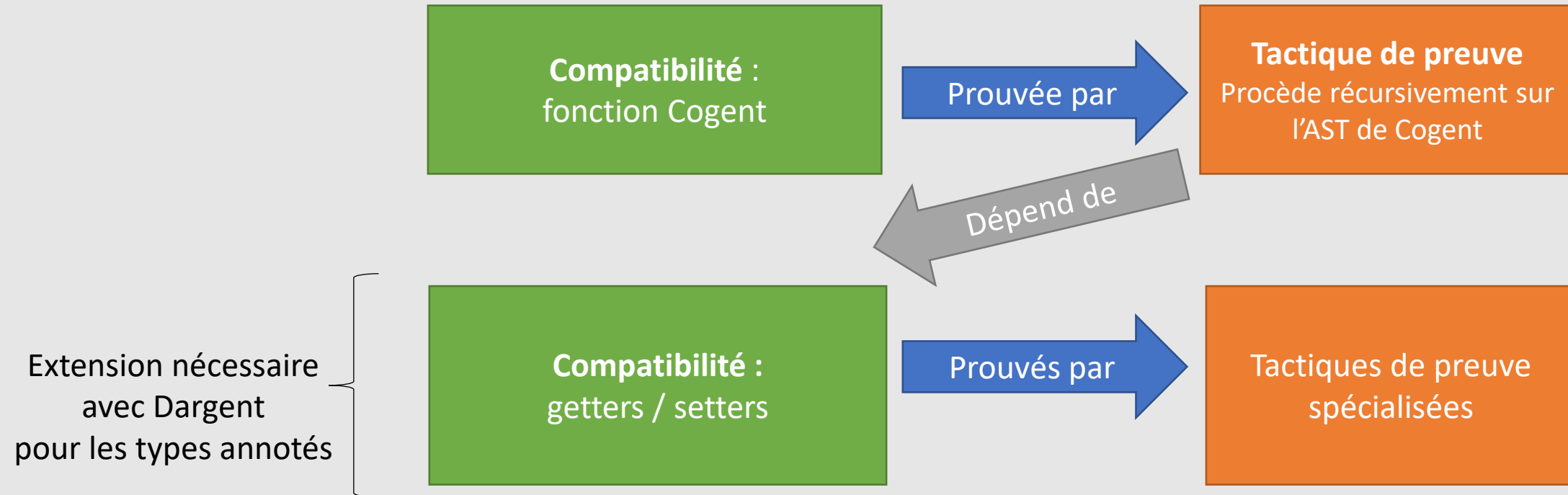
Tâche accomplie : Enoncer la compatibilité.

Sémantique AutoCorres	Sémantique simplifiée
Type T_m	Type T_s
$f_m: A_m \rightarrow B_m$	$f_s: A_s \rightarrow B_s$

Compatibilité entre les deux sémantiques
T_r = Relation de décodage (type annoté)
(f_m, f_s) est compatible avec (A_r, B_r)

Tâche restante : Implémenter une tactique de preuve pour établir la correspondance (invoquée pour chaque fonction Cogent compilée).

La preuve de compatibilité dans Cogent



Compatibilité entre les sémantiques des getters / setters

Sémantique monadique	Sémantique simplifiée
$T_m = \text{byte}[n]$	$T_s = \{ \dots, x : A, \dots \}$
get_x	$t \mapsto t.x$
set_x	$(t, a) \mapsto (t.x := a)$

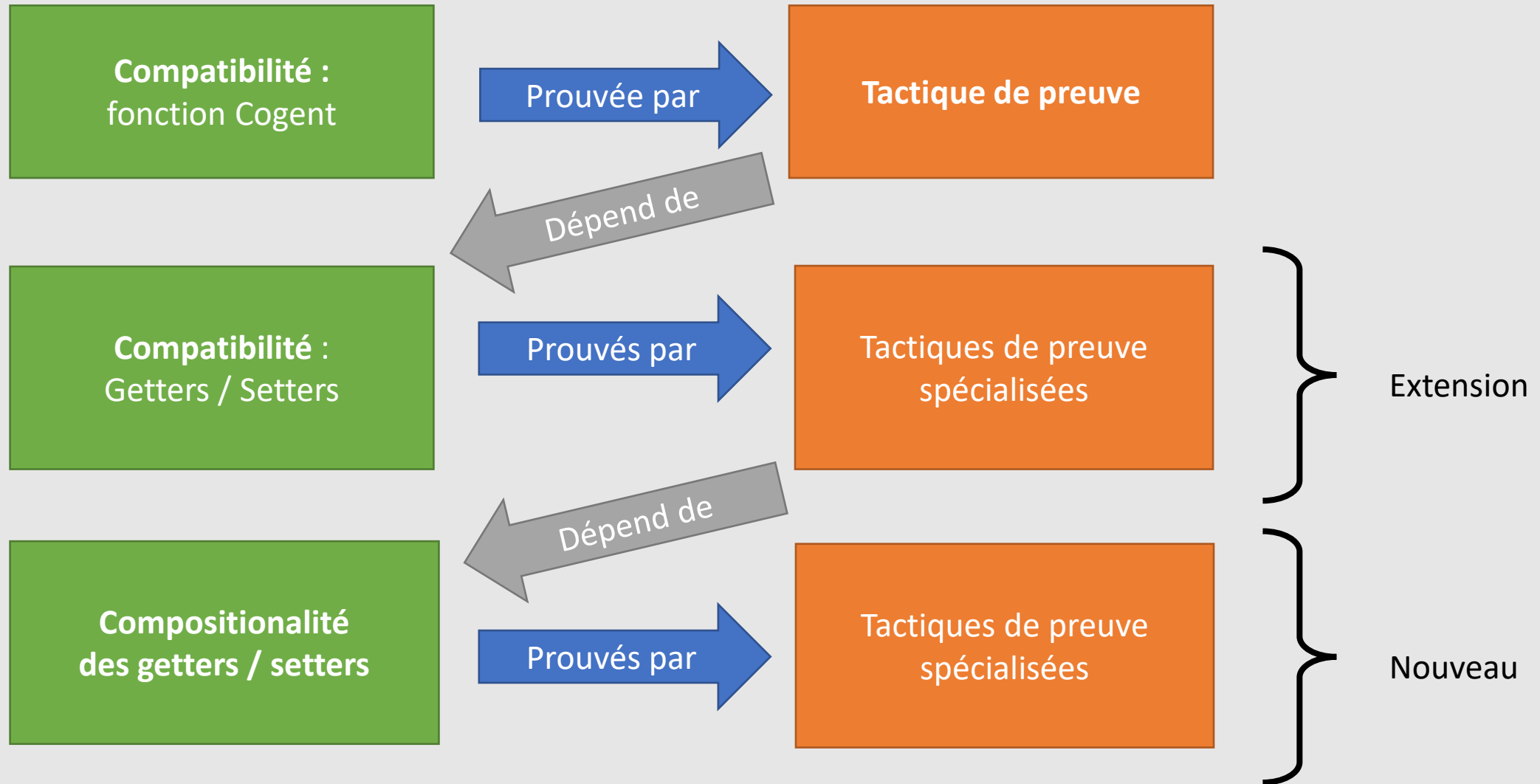
Ma méthodologie :

Quelques preuves manuelles de compatibilité pour des cas particuliers

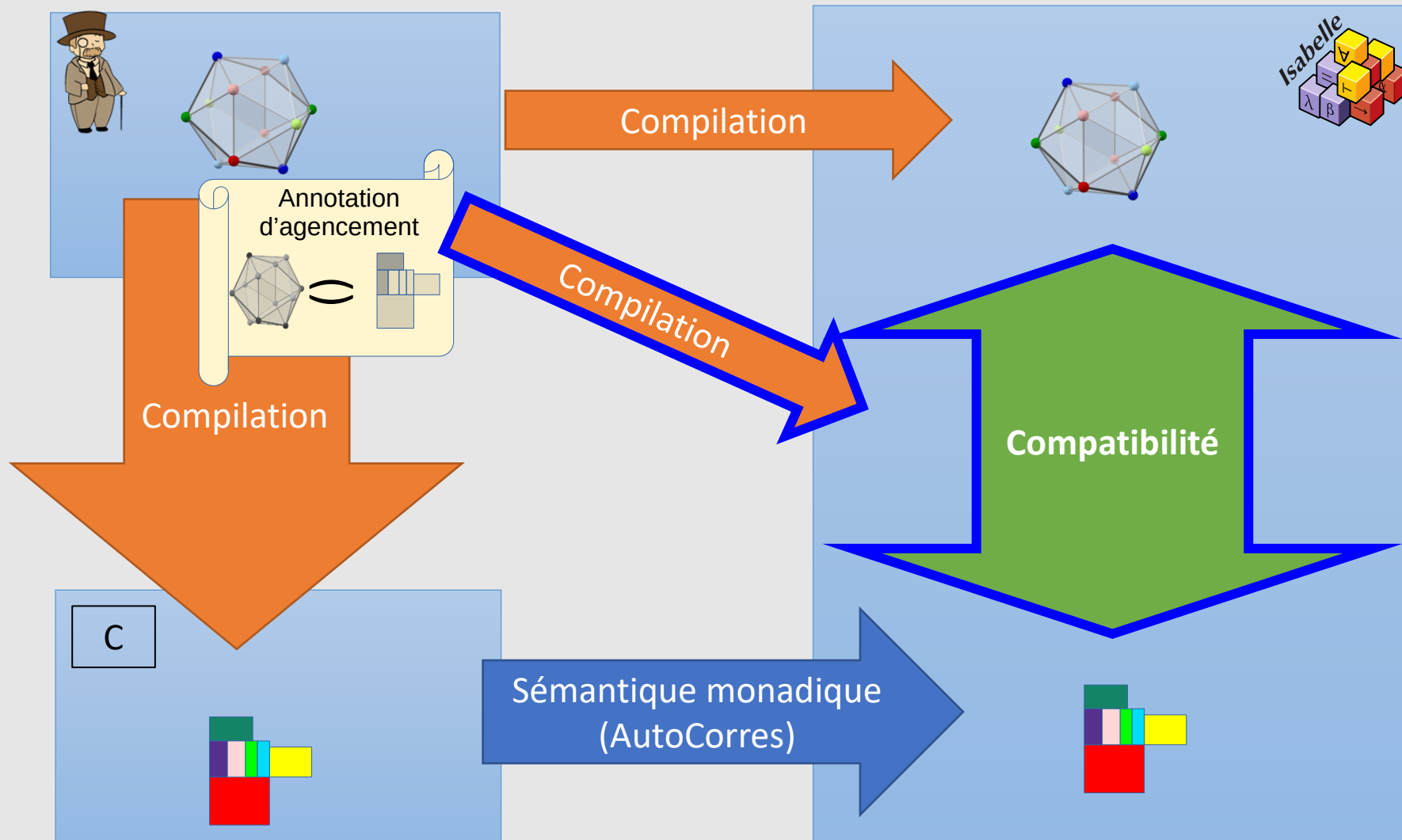
⇒ Déterminer des propriétés de compositionnalité suffisantes pour abstraire ces preuves

$$\ll \begin{array}{l} get_{\textcolor{blue}{x}}(set_{\textcolor{blue}{x}}(\text{tableau}, \text{valeur})) = \text{valeur} \\ get_{\textcolor{blue}{x}}(set_{\textcolor{red}{y}}(\text{tableau}, \text{valeur})) = get_{\textcolor{blue}{x}}(\text{tableau}) \end{array} \gg$$

Architecture finale de la preuve de compatibilité



Mission accomplie (2021)



Insuffisance de la compatibilité


Observation :

- La compatibilité ne donne aucune garantie que les getters / setters générés sont conformes à l'agencement spécifié !
- On ne savait pas formuler la conformité en question

Ce que j'ai fait

1. Une tactique de preuve pour montrer que le setter généré ne modifie que l'emplacement spécifié pour le champ
2. Un getter générique en Isabelle, paramétré par un agencement ($\in \text{TCB}$)
3. Une formulation de la condition de conformité pour les getters :
Getter généré = spécialisation du getter générique avec l'agencement spécifié
4. Une tactique de preuve pour établir cette conformité

Autres contributions sur Dargent

- Extension de la mécanisation du langage Cogent pour couvrir Dargent
⇒ Découverte & correction d'un bug dans le système de types étendu avec Dargent
- Découverte & correction d'un bug dans AutoCorres
- Extension Cogent pour des entiers de taille non standard (partie certification)

Exemple : entier sur 17 bits
- Trouver et traiter des exemples concrets d'utilisation de Dargent
En particulier : deux petits pilotes de périphérique, dont l'un vérifié formellement
Timer pour odroid, système de contrôle de puissance pour STMG4
- Article POPL 2023 (rédacteur principal de 3 sections sur 6)

[Dargent: A Silver Bullet for Verified Data Layout Refinement](#)

Chen-Lafont-O'Connor-Keller-McLaughlin-Jackson-Rizkallah

Résumé des compétences acquises

- Informatique théorique
 - Théorie des catégories (y compris supérieures)
 - Théorie des types (y compris homotopiques)
 - Théorie des langages de programmation
- Mécanisation avec assistants de preuve (Coq / Agda / Isabelle)
 - Compilation certifiée
 - Preuves de théorèmes (langages de programmation, théorie des catégories)
 - Tactiques de preuve
- Programmation
 - Editeur de diagrammes (Elm / javascript)
 - Pilotes de périphériques (Cogent / C)
 - Compilateur Cogent (Haskell)

Projet de recherche

Contexte

Métathéorie des langages de programmation :

- Des problématiques classiques

*normalisation, confluence, unification, sûreté du typage,
congruence de la bisimilarité, correction de compilateurs...*

- Deux types de contributions

(1) Adapter un résultat antérieur à un nouveau langage

Gordon '99 : congruence de la bisimilarité pour PCF

(2) Généraliser des résultats antérieurs à une classe de langages

*Plotkin-Turi '97 : congruence de la bisimilarité
pour une classe de langages du premier ordre*

- Intérêt de (1) évident !
- Intérêts de (2) ?
 - Unifier, clarifier
 - Aider à la création et à l'étude de nouveaux langages

Projet de recherche

- Une bibliothèque mécanisée en Coq pour la théorie des langages de programmation

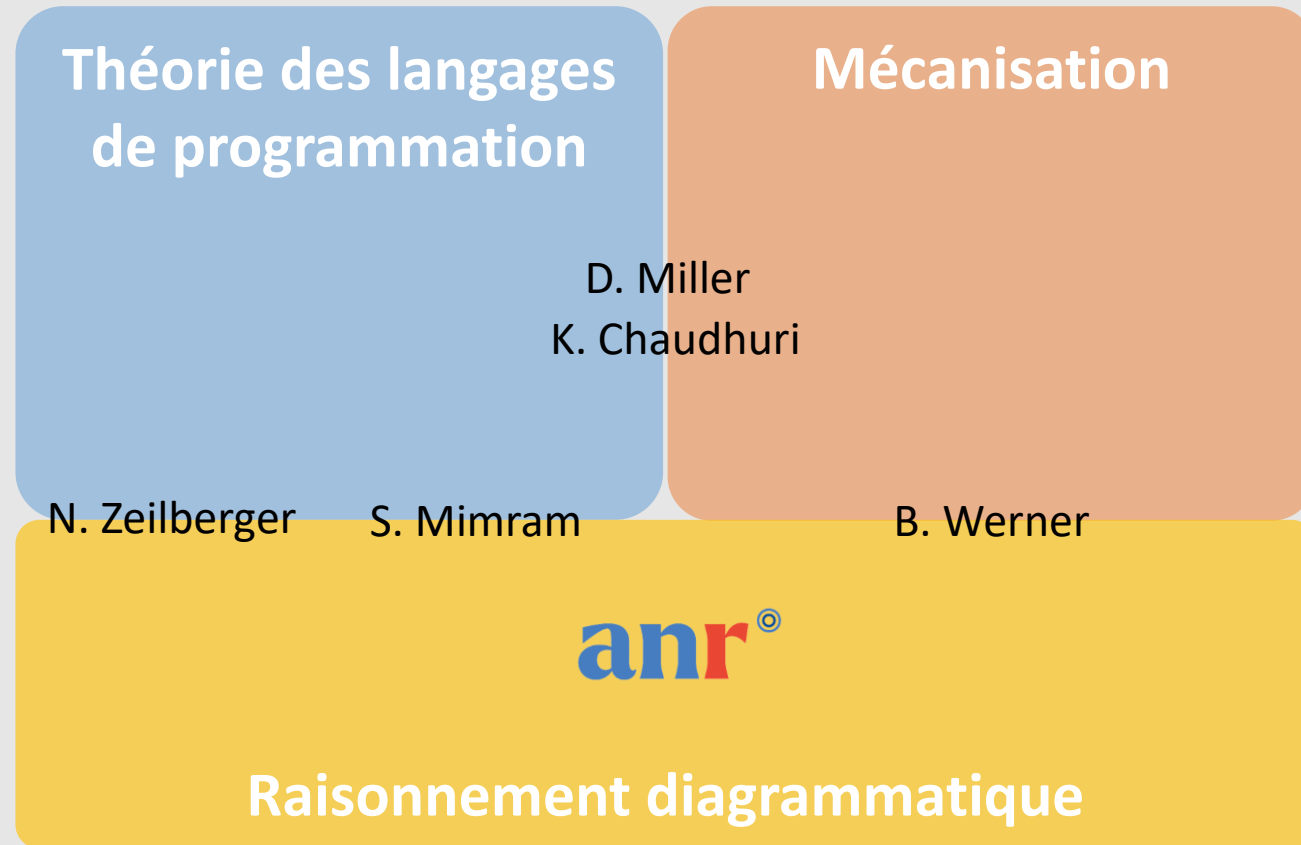
Des classes de LPs, chacune accompagnée de réponses à certaines des problématiques classiques.

- Assistants de preuve
 - Des sémantiques plus proches de l'implémentation
 - Raisonnement diagrammatique

Quelques objectifs concrets

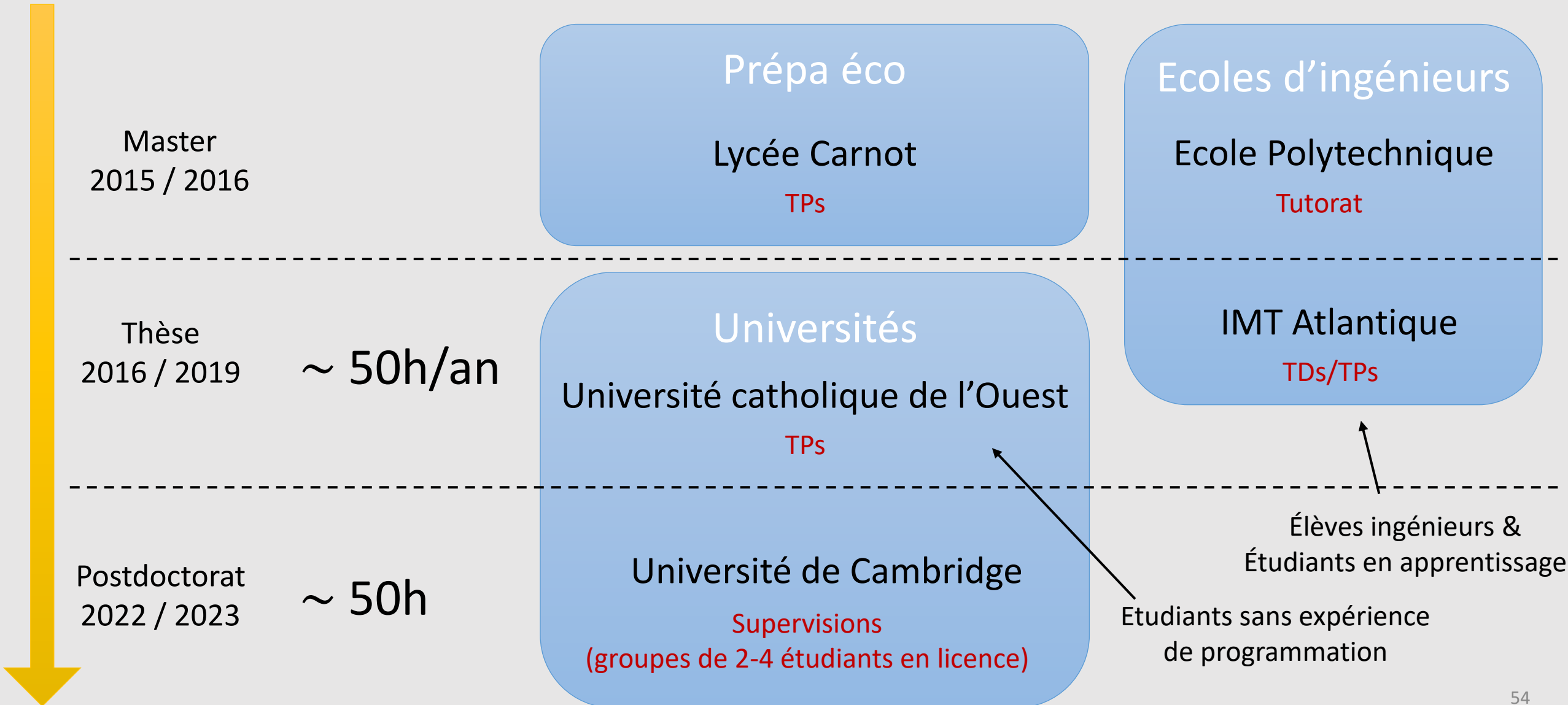
- Congruence de la bisimilarité
 - Autres notions de bisimilarité
 - Mécanisation
- Unification
 - Types dépendants
 - Modulo réduction
 - Bibliothèque certifiée
- Sûreté du typage (long terme)
- Compilation générique (long terme)

Intégration dans le pôle Preuves et Algorithmes



Enseignement

Expérience d'enseignement



Matières enseignées

Langages de programmations

Maple / Ocaml / Haskell
Python / Java

Algorithmes

- **Structures algorithmiques**
- Programmation linéaire

Informatique théorique

- Mathématiques discrètes
- Fondements de l'informatique
- Sémantique dénotationnelle

En gras : $\geq 30h$ d'enseignement

Autres :

- Probabilités & Statistiques
- Physique (relativité restreinte, mécanique quantique)

Intégration au département

Exemples de cours auxquels je pourrais contribuer facilement

Programmation

Computer Programming
Functional Programming
Web Programming
Compilers

Algorithmique

Introduction to Algorithms
Design and Analysis of algorithms

Systèmes

Computer Architecture

Mathématiques pour l'informatique

Introduction to Formal Languages
Fondements de l'informatique
Logic and Proofs

Proposition de cours

Vérification formelle de code système

- Détailler l'exemple de sel4
- Complément à INF551 qui présente les bases des assistants de preuve
- Enseignement par projets : confronter les étudiants à des problématiques réalistes

Ma philosophie de l'enseignement

Points-clés de l'apprentissage

- La pratique
- L'erreur
- L'intuition

Ma stratégie d'enseignant

- Répéter, reformuler
- Devoirs réguliers
- Exercices : trouver & corriger les erreurs
- Demander des retours sur l'enseignement
- Illustrer (exemples & contre-exemples)
- Motiver les définitions et les résultats

Résumé de mon profil

Théorie des
langages de
programmation

- Syntaxe



- Substitution
- Unification

CSL 2018, FSCD 2019, LMCS 2021, FoSSaCS 2022
Preprint 2022

- Sémantique opérationnelle

- Substitution
- Equivalences de programmes

POPL 2020, FSCD 2020, LMCS 2022
LICS 2020, LMCS 2022

Assistants de
preuve

- Fondements



- Ergonomie



TYPES 2019
Application Web

- Compilation certifiée

Conversion des clôtures

Cogent



POPL 2023