Langages de programmation et preuves mécanisées

Ambroise Lafont

Audition CNRS
24 mars 2023

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

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

Plan

1. **Profil**
2. Une contribution
3. Projet de recherche

Certification de programmes

Construction interactive

Vérification automatique

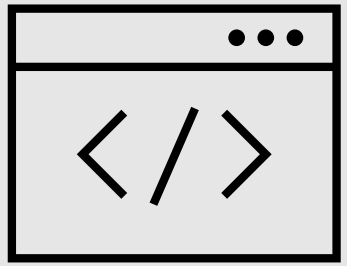
PREUVE MÉCANISÉE
(écrite dans un langage formel)

Le programme est correct

Agda

Expérience personnelle
avec 3 assistants de preuve majeurs

Programme certifié



Compilateur
gcc (**15M** de lignes)

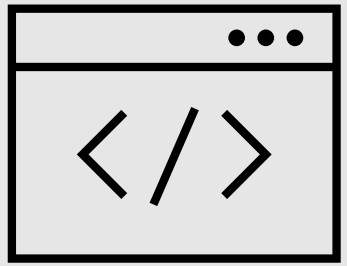
Exécutable

Règle empirique [1] : 1 bug / 1k lignes

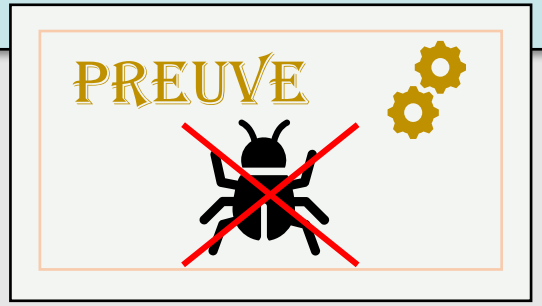
[1] McConnell, “Code Complete: A Practical Handbook of Software Construction”

Programme certifié

Exécutable

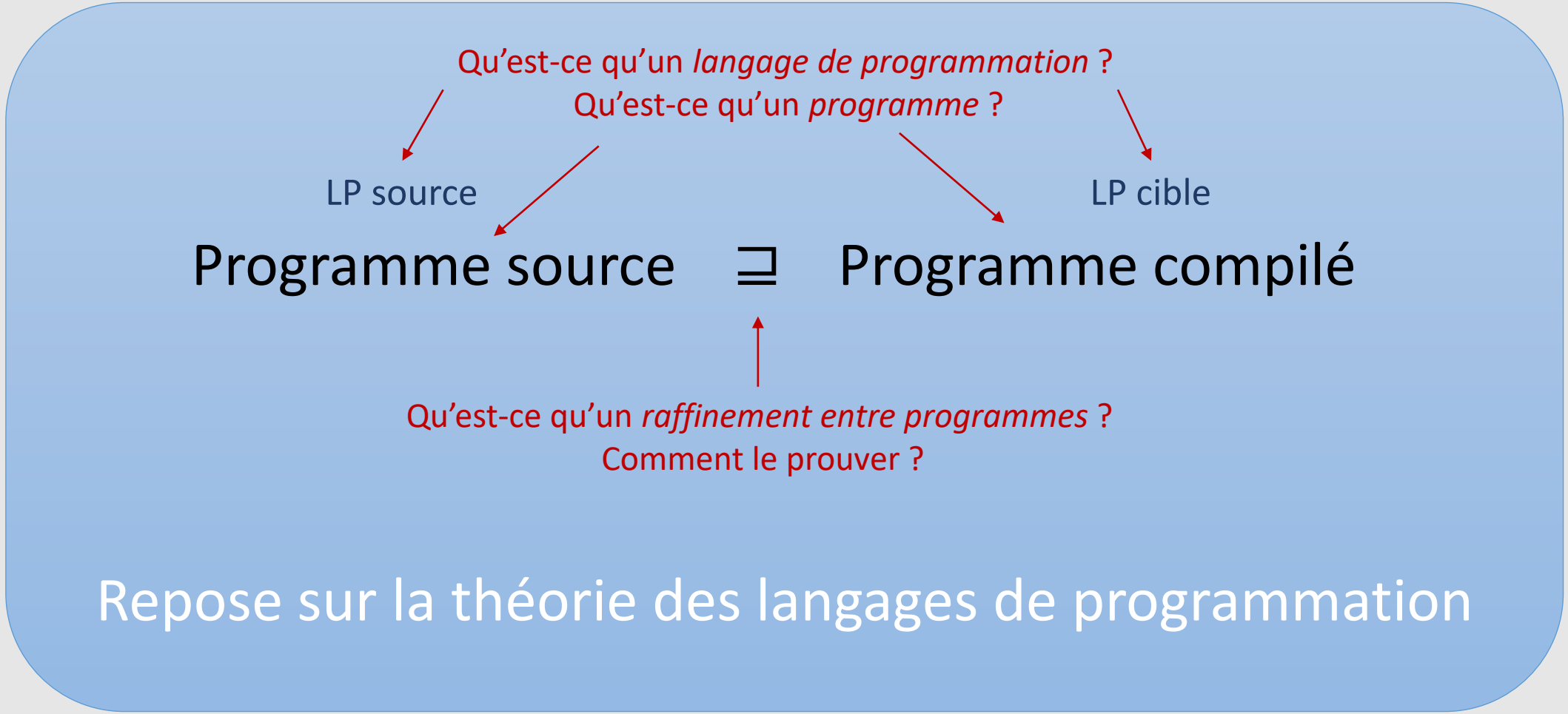


Compilateur
certifié



100k lignes

Correction d'un compilateur

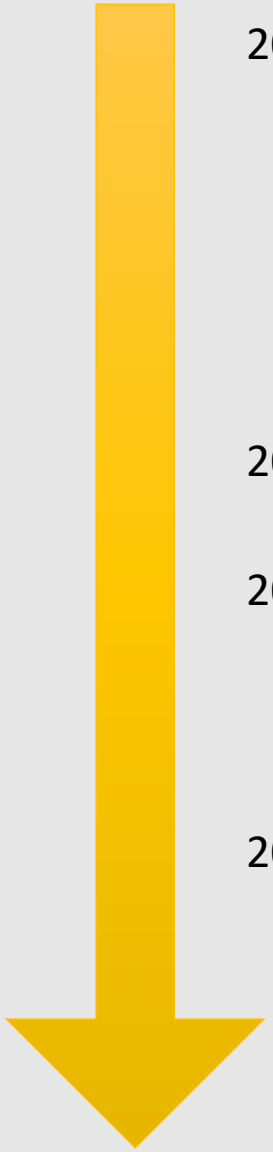


“What’s a program? (Seriously)”

Liste de diffusion TYPES (2021)

- [TYPES] What's a program? (Seriously) *Talia Ringer*
  - [TYPES] What's a program? (Seriously) *Stefan Monnier*
    - [TYPES] What's a program? (Seriously) *Gavin Mendel-Gleason*
  - [TYPES] What's a program? (Seriously) *Sergey Goncharov*
  - [TYPES] What's a program? (Seriously) *Martin Escardo*
    - [TYPES] What's a program? (Seriously) *Martin Escardo*
      - [TYPES] What's a program? (Seriously) *Talia Ringer*
      - [TYPES] What's a program? (Seriously) *Martin Escardo*
      - [TYPES] What's a program? (Seriously) *Hendrik Boom*
      - [TYPES] What's a program? (Seriously) *Martin Escardo*
  - [TYPES] What's a program? (Seriously) *Neel Krishnaswami*
    - [TYPES] What's a program? (Seriously) *Sandro Stucki*
      - [TYPES] What's a program? (Seriously) *Neel Krishnaswami*
      - [TYPES] What's a program? (Seriously) *Talia Ringer*
      - [TYPES] What's a program? (Seriously) *Jason Gross*
      - [TYPES] What's a program? (Seriously) *Gabriel Scherer*
      - [TYPES] What's a program? (Seriously) *Thomas Streicher*
      - [TYPES] What's a program? (Seriously) *Guillaume Munch-Maccagnoni*
      - [TYPES] What's a program? (Seriously) *Tadeusz Litak*
      - [TYPES] What's a program? (Seriously) *Jason Gross*
      - [TYPES] What's a program? (Seriously) *Gabriel Scherer*
      - [TYPES] What's a program? (Seriously) *Thomas Streicher*
      - [TYPES] What's a program? (Seriously) *Neel Krishnaswami*
      - [TYPES] What's a program? (Seriously) *Tarmo Uustalu*
      - [TYPES] What's a program? (Seriously) *matthias at ccs.neu.edu*
      - [TYPES] What's a program? (Seriously) *Oleg*
      - [TYPES] What's a program? (Seriously) *Guillaume Munch-Maccagnoni*
      - [TYPES] What's a program? (Seriously) *Nicolai Kraus*
      - [TYPES] What's a program? (Seriously) *Thomas Streicher*
      - [TYPES] What's a program? (Seriously) *Oleg*
    - [TYPES] What's a program? (Seriously) *Ansten Mørch Klev*
      - [TYPES] What's a program? (Seriously) *Freek Wiedijk*
    - [TYPES] What's a program? (Seriously) *Tom Hirschowitz*
      - [TYPES] What's a program? (Seriously) *Hendrik Boom*
  - [TYPES] What's a program? (Seriously) *streicher at mathematik.tu-darmstadt.de*
    - [TYPES] What's a program? (Seriously) *Guillaume Munch-Maccagnoni*
  - [TYPES] What's a program? (Seriously) *Andrew Polonsky*

Parcours académique



2016

Stage de Master avec X. Leroy, **Paris**
Phase de compilation *certifiée*

Compilateur fiable

2019

PhD avec N. Tabareau & T. Hirschowitz, **LS2N (Nantes)**
Signatures et modèles pour la syntaxe et la sémantique opérationnelle en présence de liaison de variables

Théorie des langages de programmation

2020

Postdoc avec C. Rizkallah, **UNSW (Sydney)**
Compilateur *certifié* pour la programmation système

Compilateur fiable

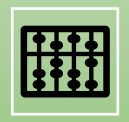
2022

Postdoc avec N. Krishnaswami, **University of Cambridge**
Théorie de l'*unification* (*HO*)

Théorie des langages de programmation

Publications

- Syntaxes avec substitution
- Contributions à UniMath (théorie des catégories)



Notion de langages de programmation[1]

Spécification de la syntaxe et exécution

FoSSaCS 2022, **LMCS** 2021 & 2022, **POPL** 2020, **FSCD** 2020 & 2019, **CSL** 2018

Équivalences de programmes[2]

Méthode de Howe générique

LMCS 2022, **LICS** 2020

Théorie des types[3]

Construction de types inductifs avancés

TYPES 2019

Compilateur certifié pour la programmation système

POPL 2023

[1] avec B. Ahrens, A. Hirschowitz, T. Hirschowitz, M. Maggesi
[2] avec P. Borthelle, T. Hirschowitz
[3] avec A. Kaposi, A. Kovács
[4] avec l'équipe Cogent

Contribution de mon postdoctorat

Compilateur certifié pour la programmation système

POPL 2023

Le groupe Trustworthy Systems

seL4

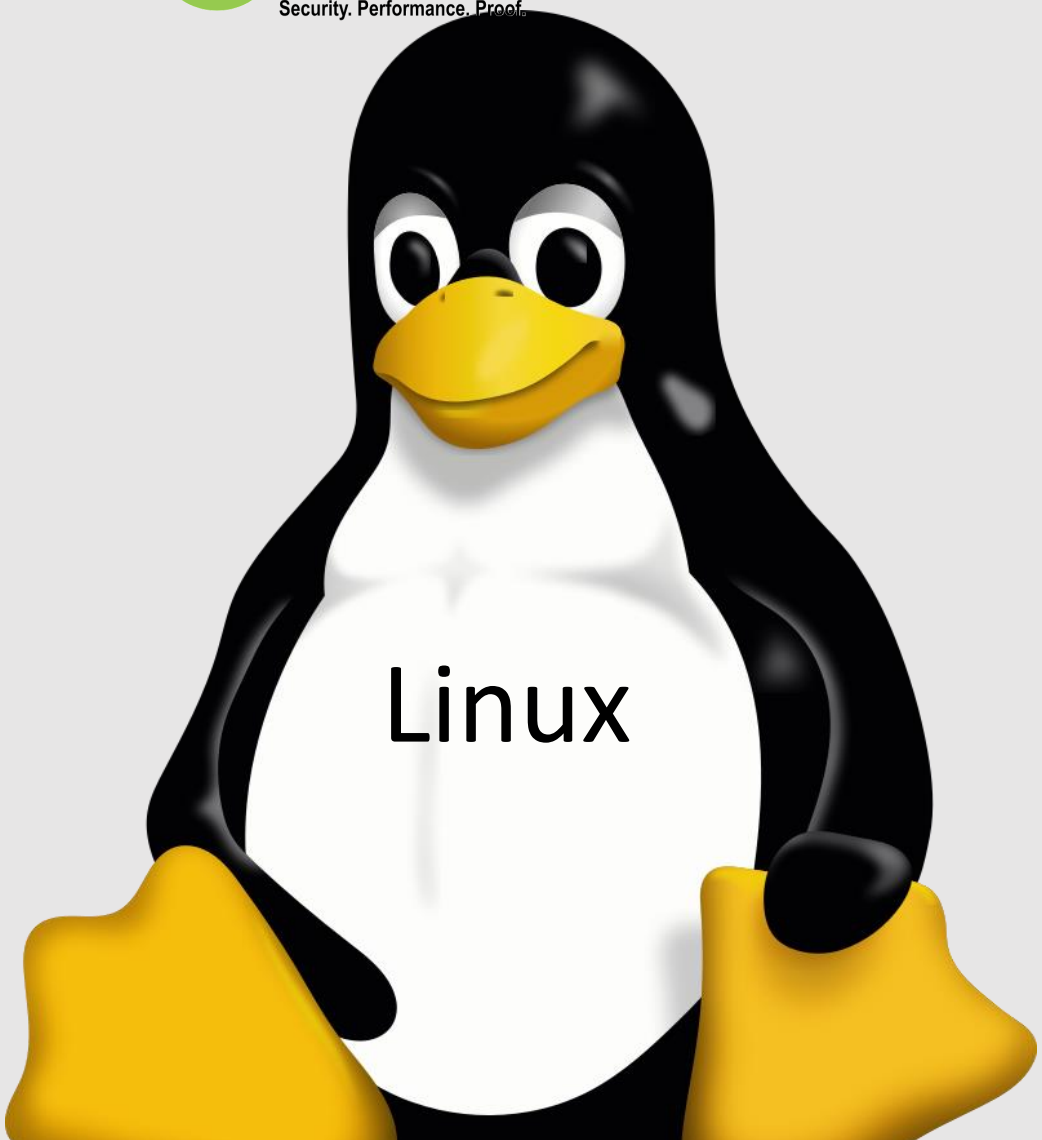
Security. Performance. Proof.

Le noyau de système d'exploitation le plus sûr au monde.

Isabelle

Utilisé dans des
véhicules autonomes

Micro-noyau: 10k lignes de C



PREUVE

200k lignes

Isabelle

30M lignes de C

Systèmes de fichiers, pilotes de périphérique, …

Comment faciliter la certification de code système ?

Thème de mon postdoctorat

Cogent : réduire le coût de la certification

Cogent : un langage fonctionnel restreint avec un compilateur générant une preuve mécanisée reliant le programme C produit et sa représentation mathématique.

Programme C

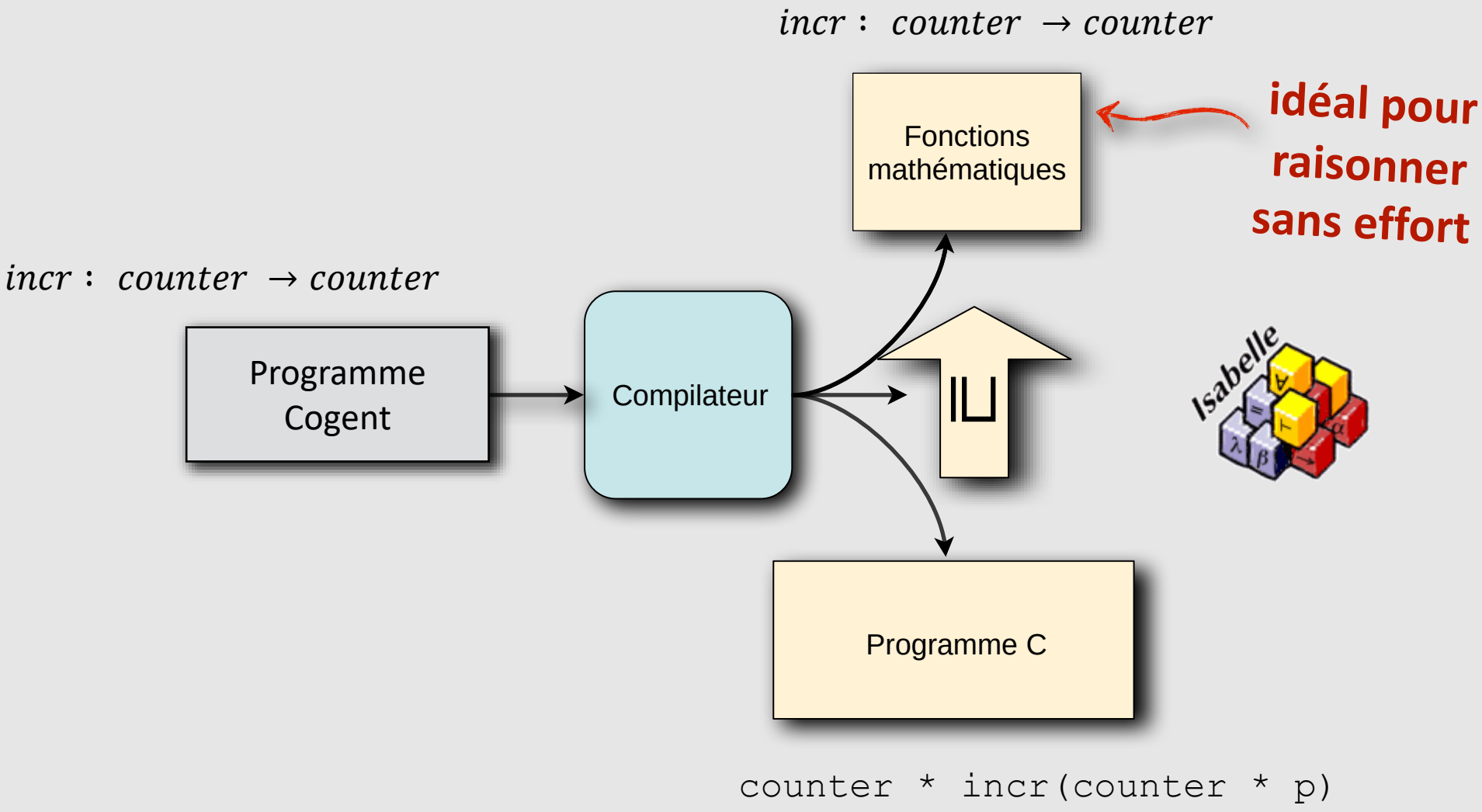
Fonction mathématique

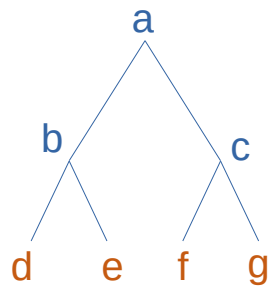
```
counter * incr(counter * p) {
  p->count = p->count + 1;
  return p;
}
```

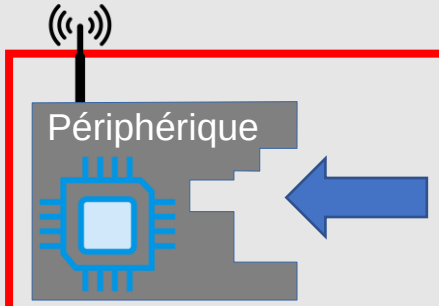
$$incr : counter \rightarrow counter$$
$$p \mapsto \{ count = p.count + 1 \}$$

Incrémente le champ `count`
à l'adresse mémoire `p`, renvoie `p`

Plus facile :
Pas de pointeur / mémoire / comportement indéfini


$incr : counter \rightarrow counter$
Fonctions mathématiques
idéal pour raisonner sans effort
$incr : counter \rightarrow counter$
Programme Cogent
Compilateur
Isabelle
Programme C
counter * incr(counter * p)

	Implementation		Verification
Code	Programmes C		Fonctions mathématiques
Données	Bits et octets en mémoire a b c d e f g Personnalisation ? c e a d g b f	$\cong$	Types algébriques 



Mon travail postdoctoral

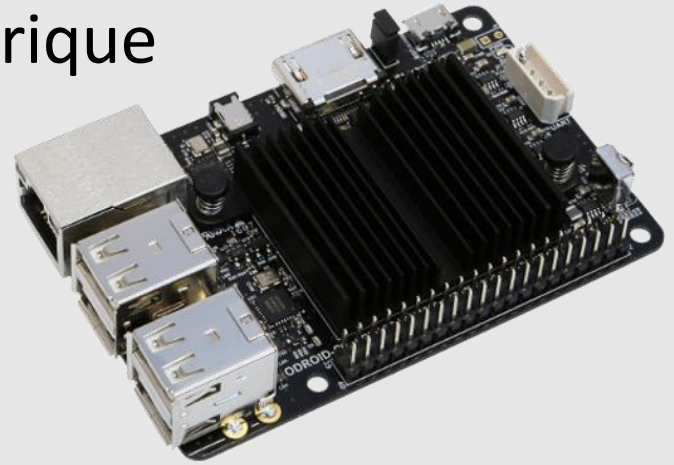
Dargent

Dargent est un langage certifié de spécification d’agencement mémoire des données.

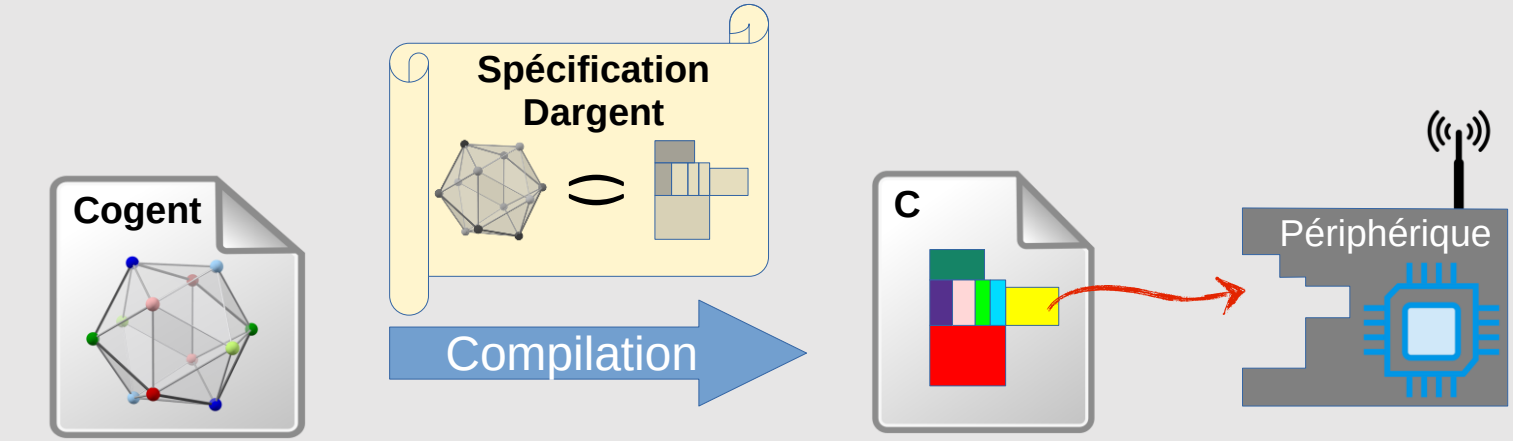
Preuve de concept (contribution personnelle)

Certification d’un petit pilote de périphérique

(timer pour odroid)



Dargent : une extension de Cogent

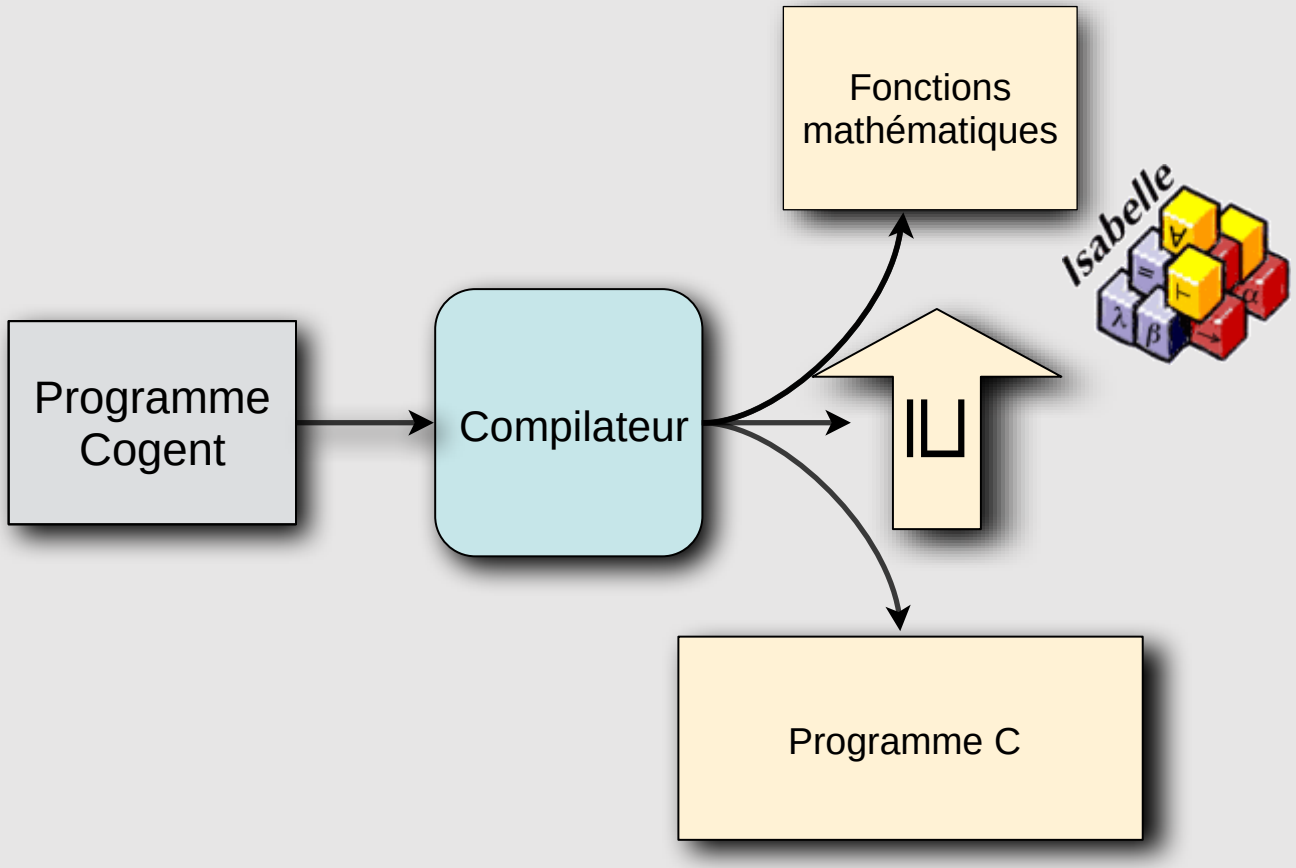


Types algébriques

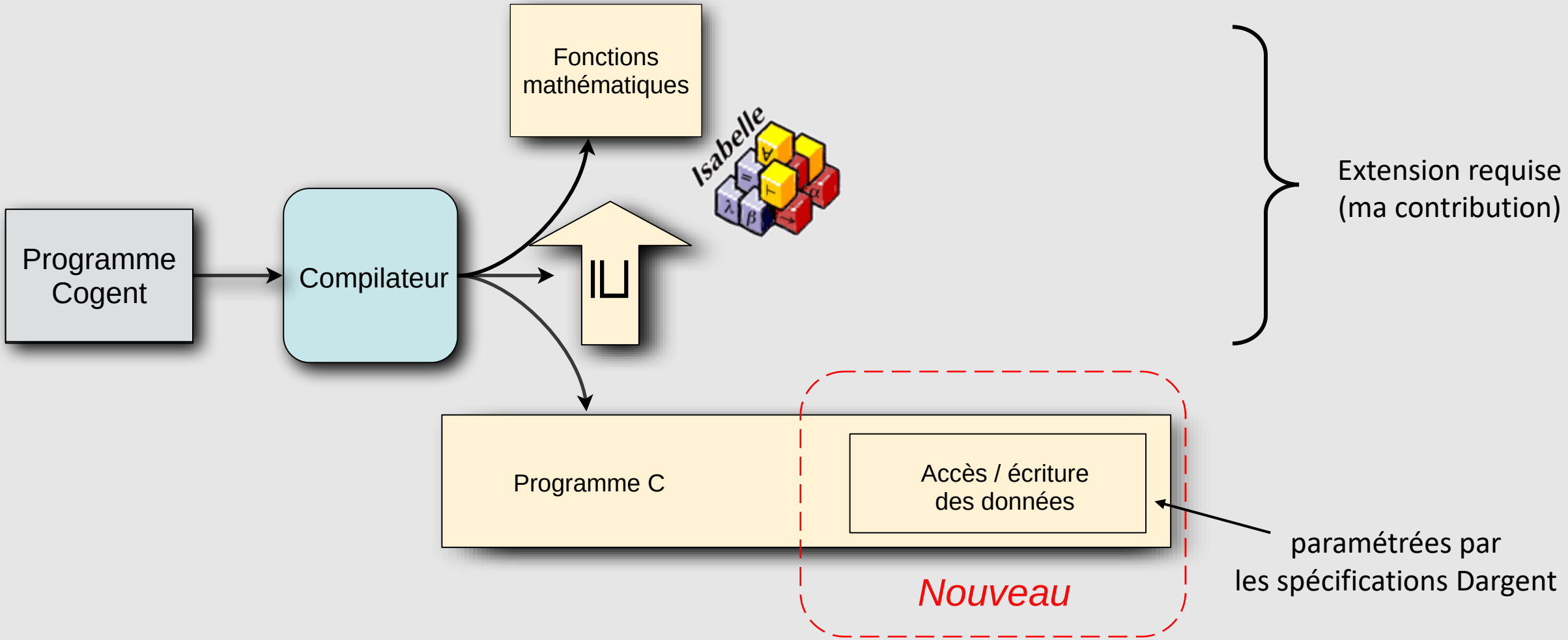
Bits et octets

\+

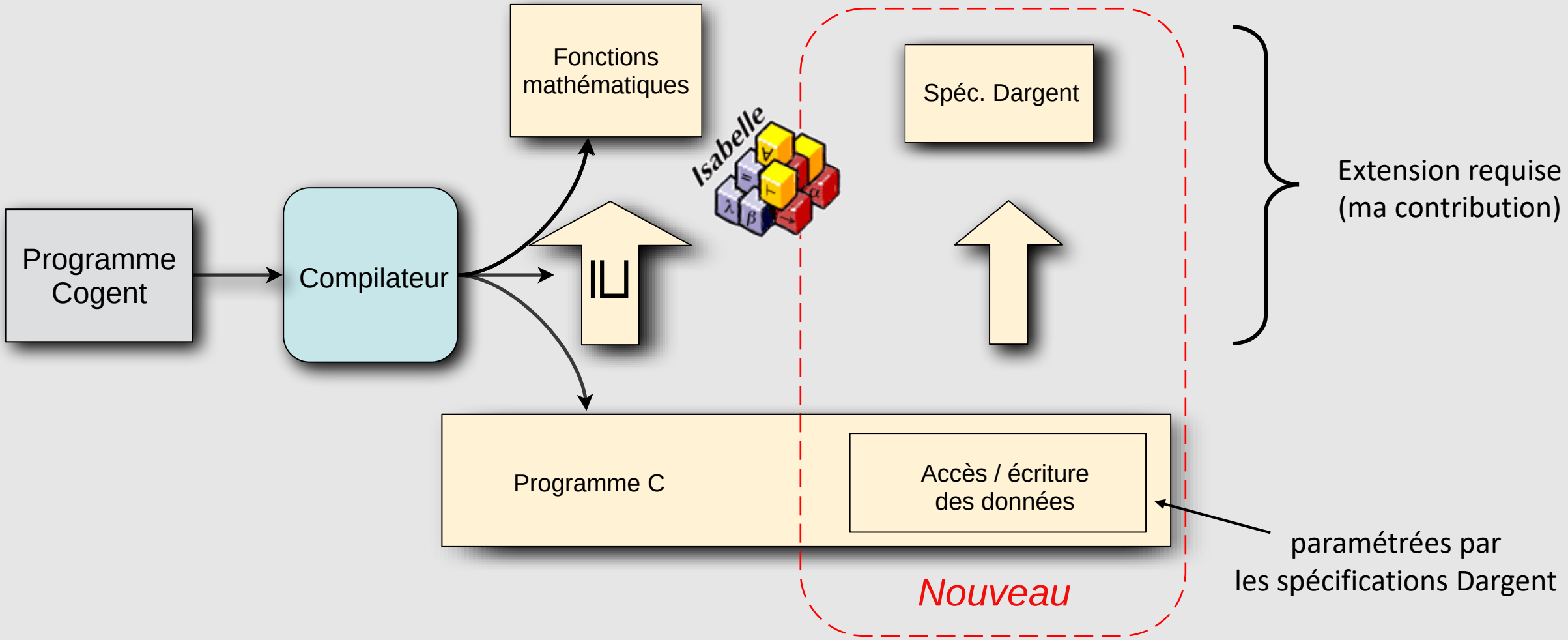
Dargent : une extension de Cogent



Dargent : une extension de Cogent



Dargent : une extension de Cogent



Cogent + Dargent

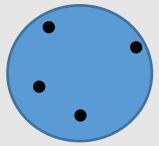
Premier langage de programmation certifié avec un contrôle de l’agencement de types compilés en mémoire.

Publication à **POPL** 2023

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

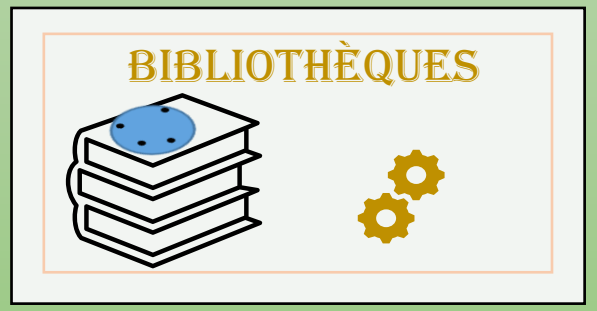
Projet de recherche

Abstraction



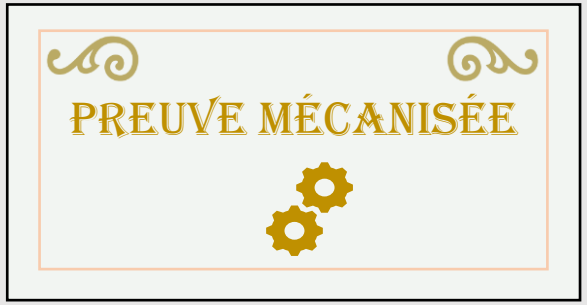
Théorème pour une classe de langages

Projet de recherche



Un langage particulier

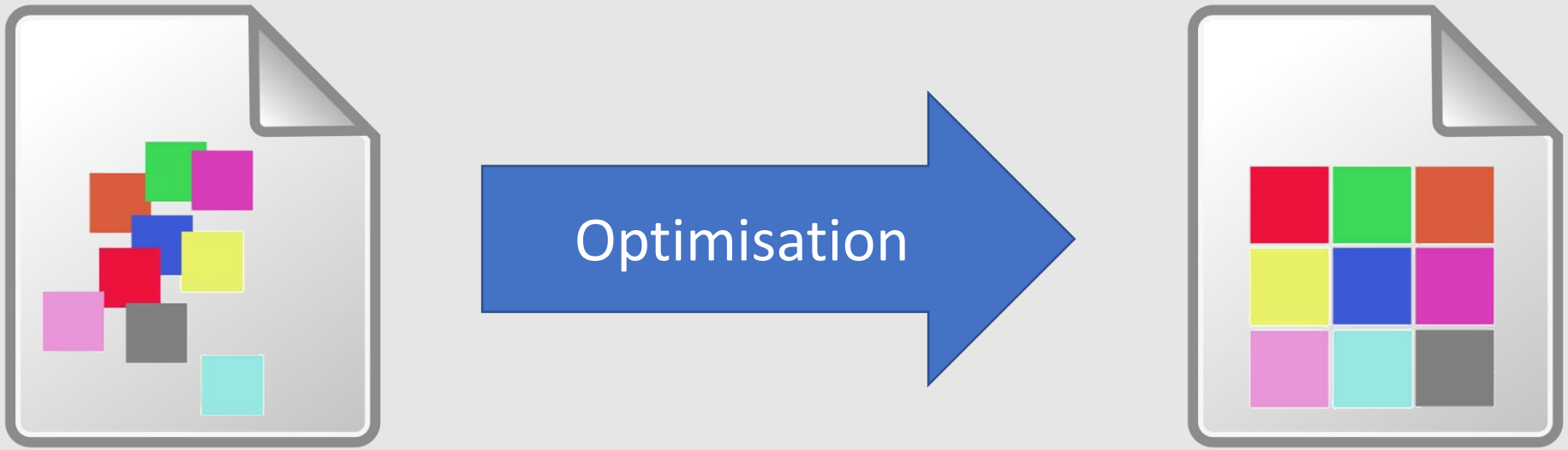
Fastidieux :
beaucoup de cas



Mécanisation

Théorie modulaire des langages de programmation

Equivalences de programmes



Contextuellement équivalents

i.e., donnent les mêmes résultats…

… dans n'importe quel contexte d'exécution

Difficile à prouver

Une solution : *passer par une autre notion d'équivalence* ~

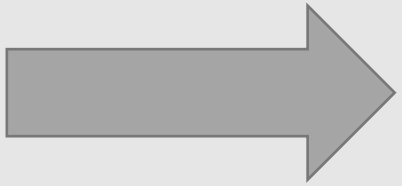
On veut ~ tel que

-  ~  facile à prouver

- ~ implique l'équivalence contextuelle

Méthode de Howe (1996) : technique de preuve pour ~ = *bisimilarité applicative*

Abstraction

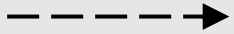


Mécanisation

[1] [2]

Méthode
de Howe

[3]



[1] [2]

Théorème
de Howe

[3]

Axe de recherche en cours
(**LICS** 2019, **LMCS** 2022)

[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

Implémentation modulaire certifiée des langages de programmation

Unification

Unification

- Programmation logique (λ-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)

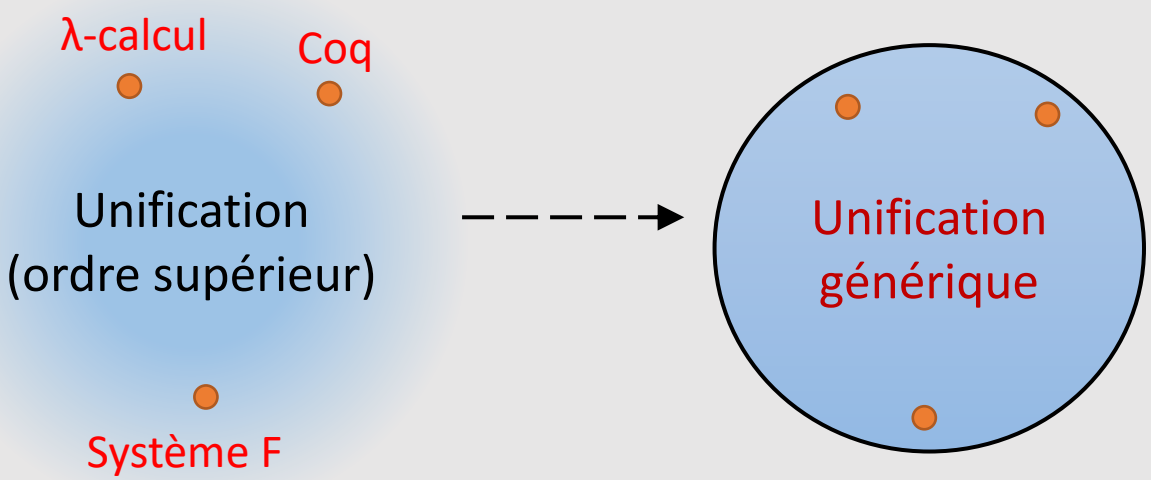
- Qui est P ?
  $x \mapsto x + 0 = x$
- Qui est n ?
  a

Indécidable

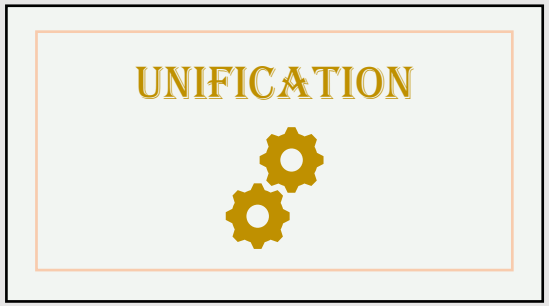
Fragment décidable identifié par Miller (1991) pour le λ-calcul

Unification

Abstraction



Mécanisation



Axe de recherche en cours

(preprint sur l'unification à la Miller, soumis avec Neel Krishnaswami)

Ergonomie des assistants de preuve

Réduire l'écart entre les mathématiques dans les assistants de preuve et les mathématiques non formalisées

Mécanisation de diagrammes

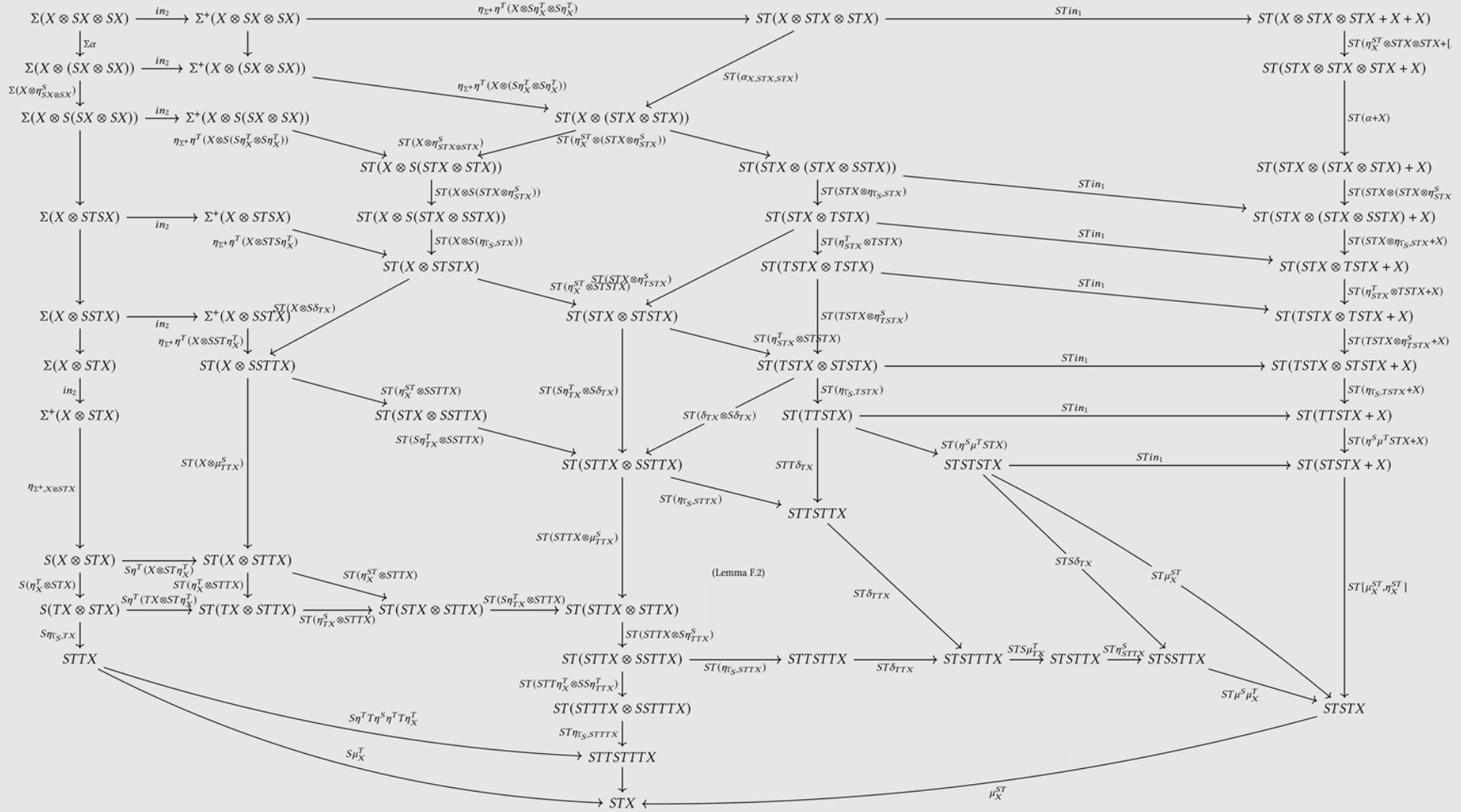
une preuve ⟶

(WIP sur la méthode de Howe)

?

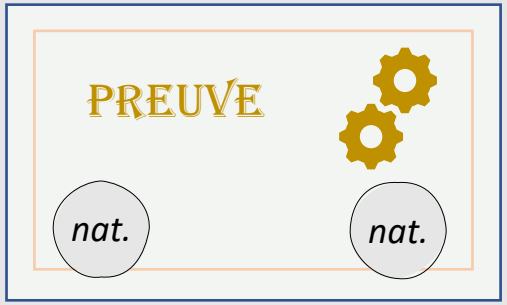
PREUVE MÉCANISÉE

texte



Enoncé

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

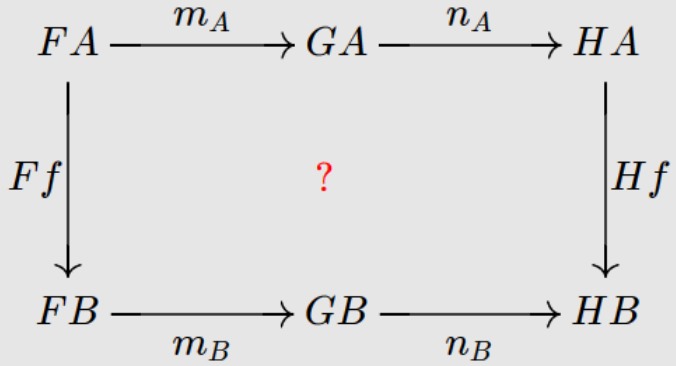


Affiche

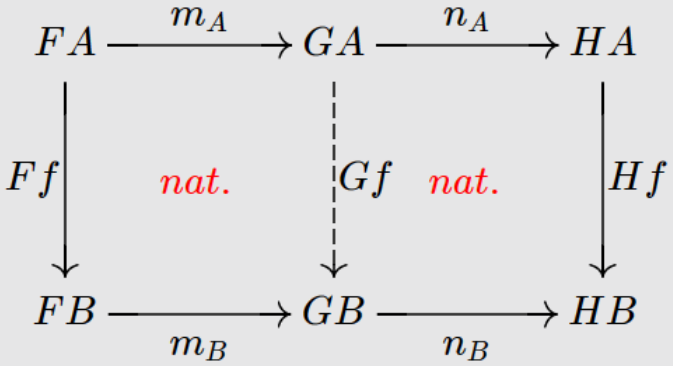
Génère

Éditeur de diagrammes

(présenté au groupe
de travail GReTA-ExACT)



Actions de l'utilisateur



Projet ANR CoREACT (2023-2027)

porté par Nicolas Behr, IRIF

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

Applications : mécanisation du théorème de Howe, …

Intégration

	Théorie des langages de programmation	Mécanisation	Raisonnement diagrammatique
LoVe, LIPN (Villetaneuse)	G. Manzonetto D. Mazza T. Seiller	M. Kerjean M. Mayero	
Plume, LIP (Lyon)	M. Mio V. Vignudelli	Y. Zakowski D. Hirschkoff D. Pous	R. Harmer
PPS, IRIF (Paris)	T. Ehrhard D. Petrisan	P. Letouzey H. Herbelin	N. Behr A. Gheerbrant P.-A. Melliès