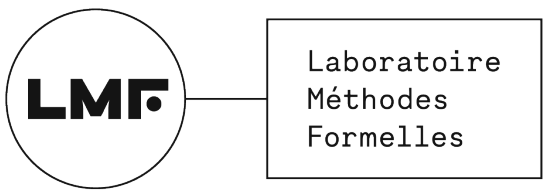
La vérification déductive avec l'outil Why3

Jean-Christophe Filliâtre
CNRS

Journée Algorithmes d'aide à la décision publique
4 octobre 2022



une approche mathématique de la correction du logiciel :

1. écrire une spécification dans un langage mathématique
2. construire une preuve que le programme la satisfait

un exemple (extrême)

```
a[52514],b,c=52514,d,e,f=1e4,g,h;main(){for(;b=c-=14;h=printf("%04d",
e+d/f))for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

un exemple (extrême)

```
a[52514],b,c=52514,d,e,f=1e4,g,h;main(){for(;b=c-=14;h=printf("%04d",
e+d/f))for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

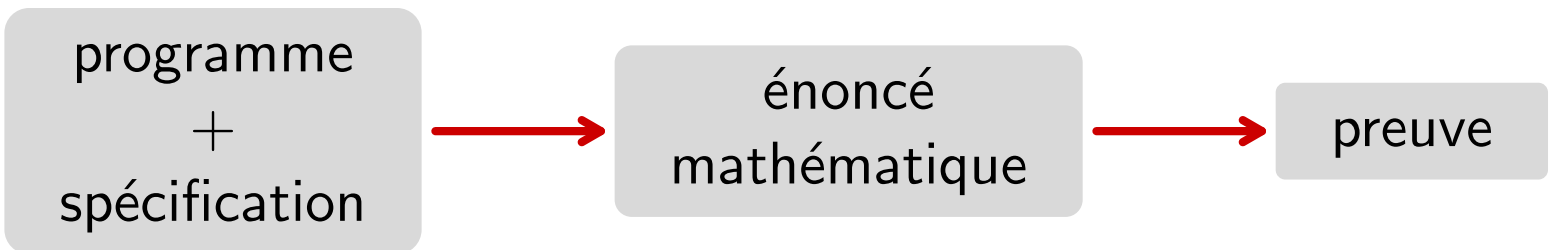
spécification affiche 15 000 décimales de π

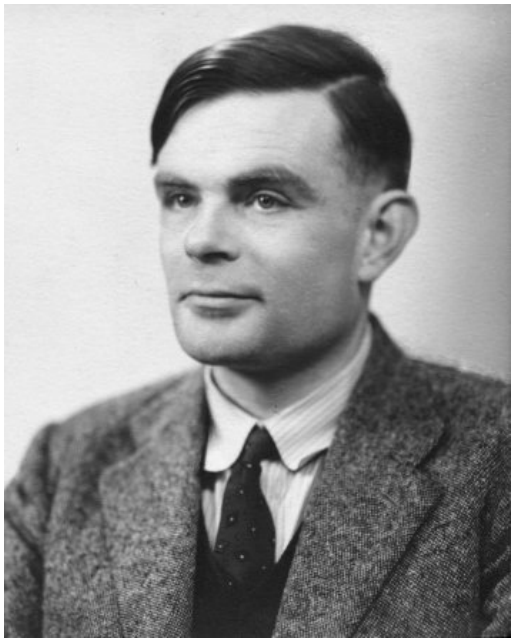
preuve beaucoup de maths, dont

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2\, 2^{i+1}}{(2i+1)!}$$

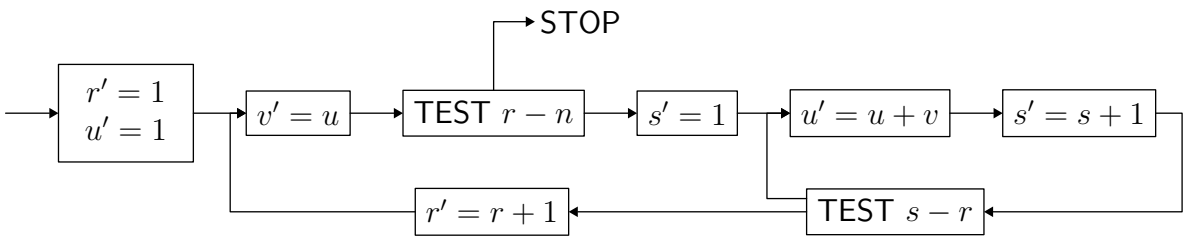
plan

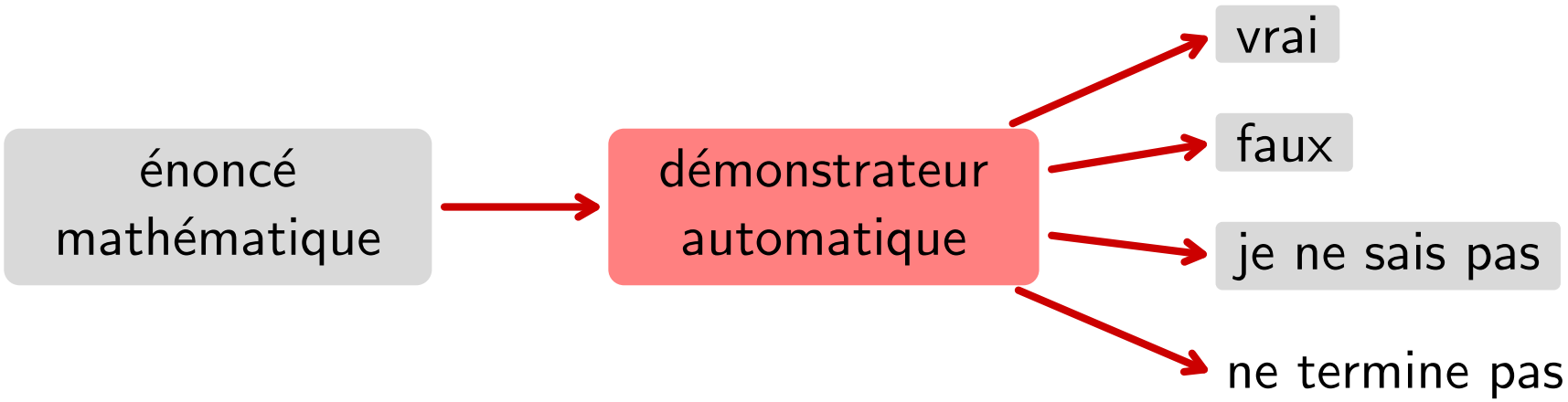
1. la vérification déductive
2. l'outil Why3
3. les enjeux de la vérification
4. leçons d'une expérience avec Parcoursup

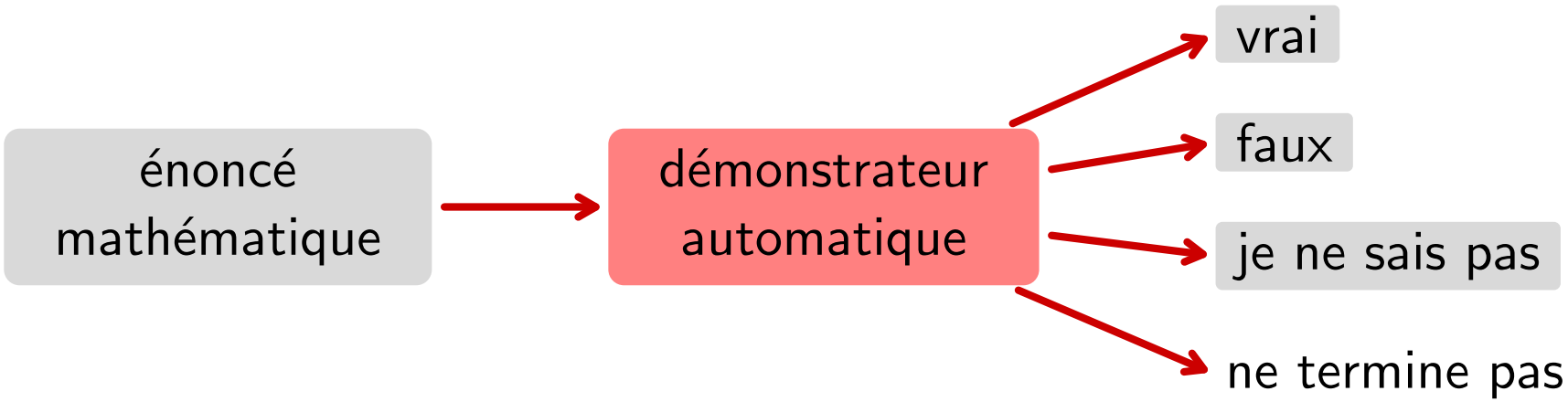

programme
+
spécification
énoncé
mathématique
preuve

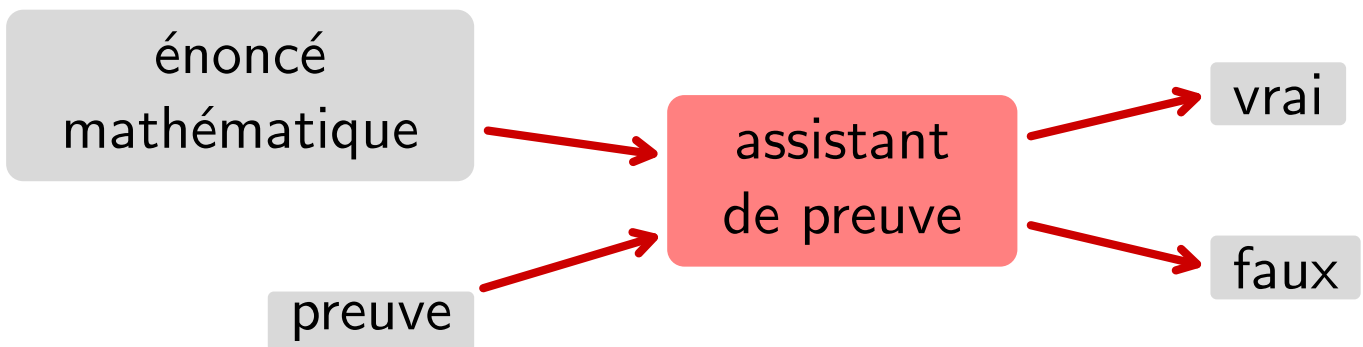


A. M. Turing. Checking a large routine. 1949.



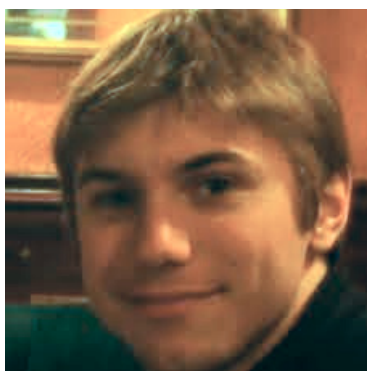
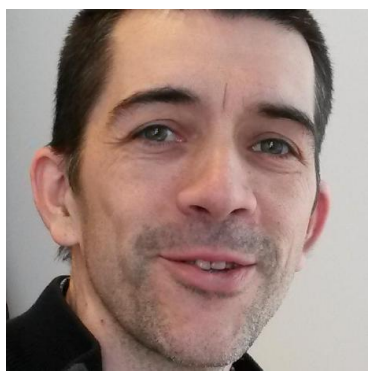

énoncé mathématique
démonstrateur automatique
vrai
faux
je ne sais pas
ne termine pas


énoncé mathématique
démonstrateur automatique
vrai
faux
je ne sais pas
ne termine pas

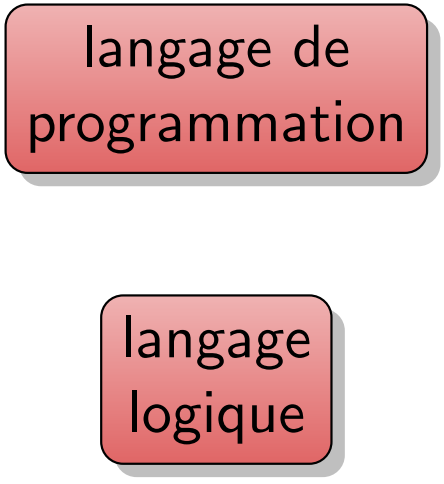

énoncé mathématique
preuve
assistant de preuve
vrai
faux

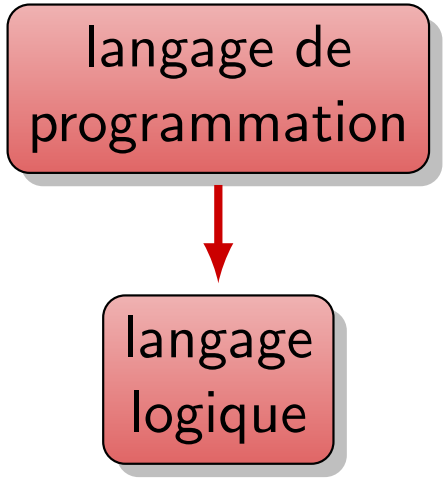
une plateforme pour la vérification déductive

- 1999 : prototype intégré à Coq
- 2001 : outil indépendant
- 2009 : refonte complète

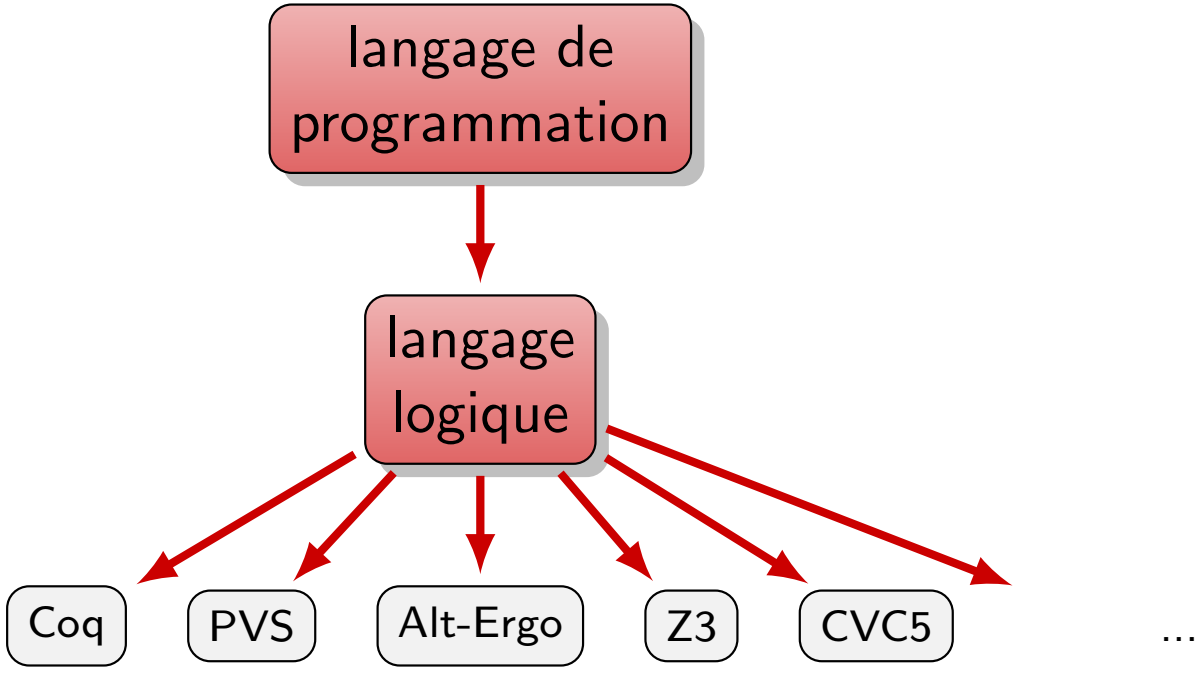




Why3, un couteau suisse

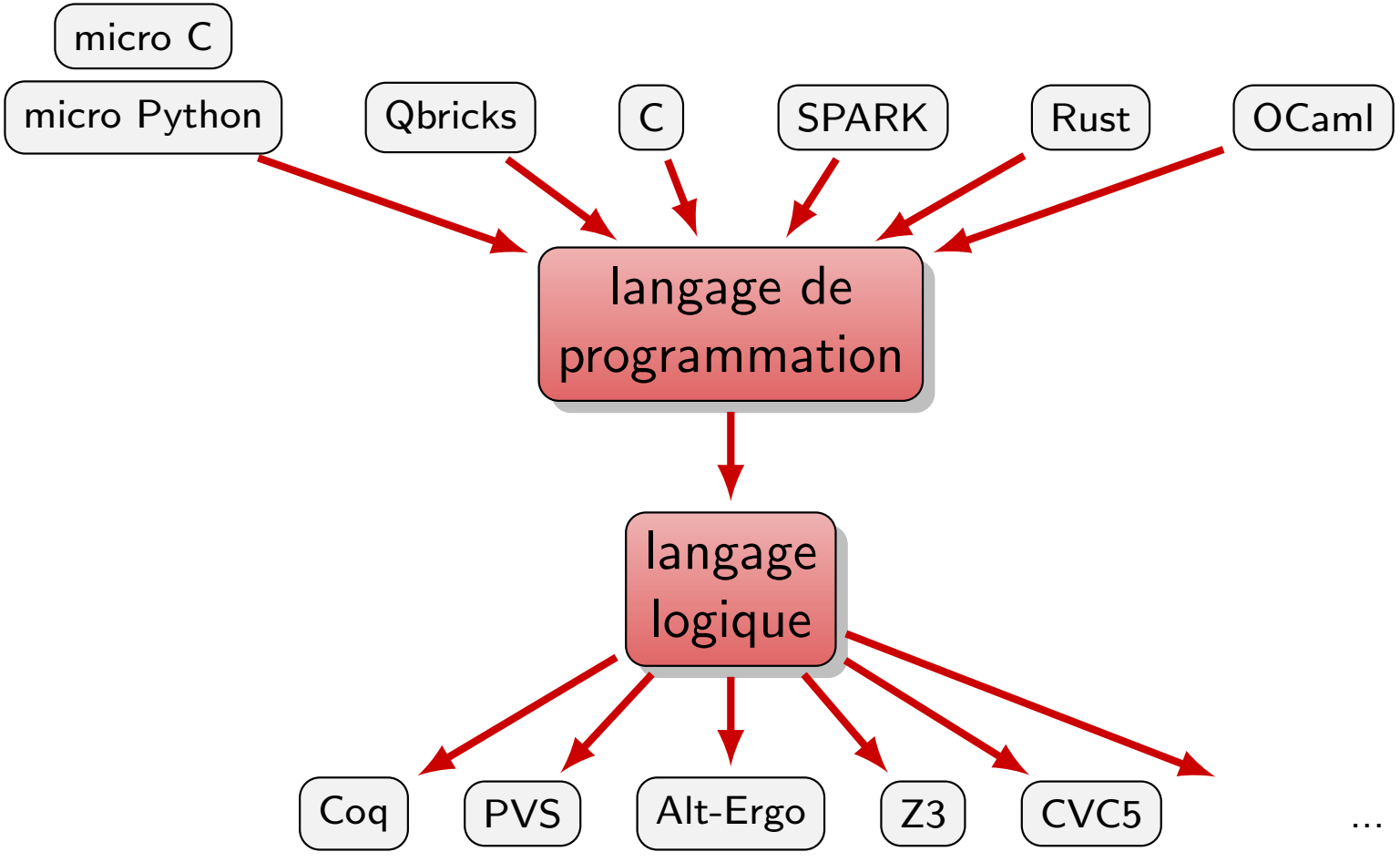



langage de
programmation
langage
logique

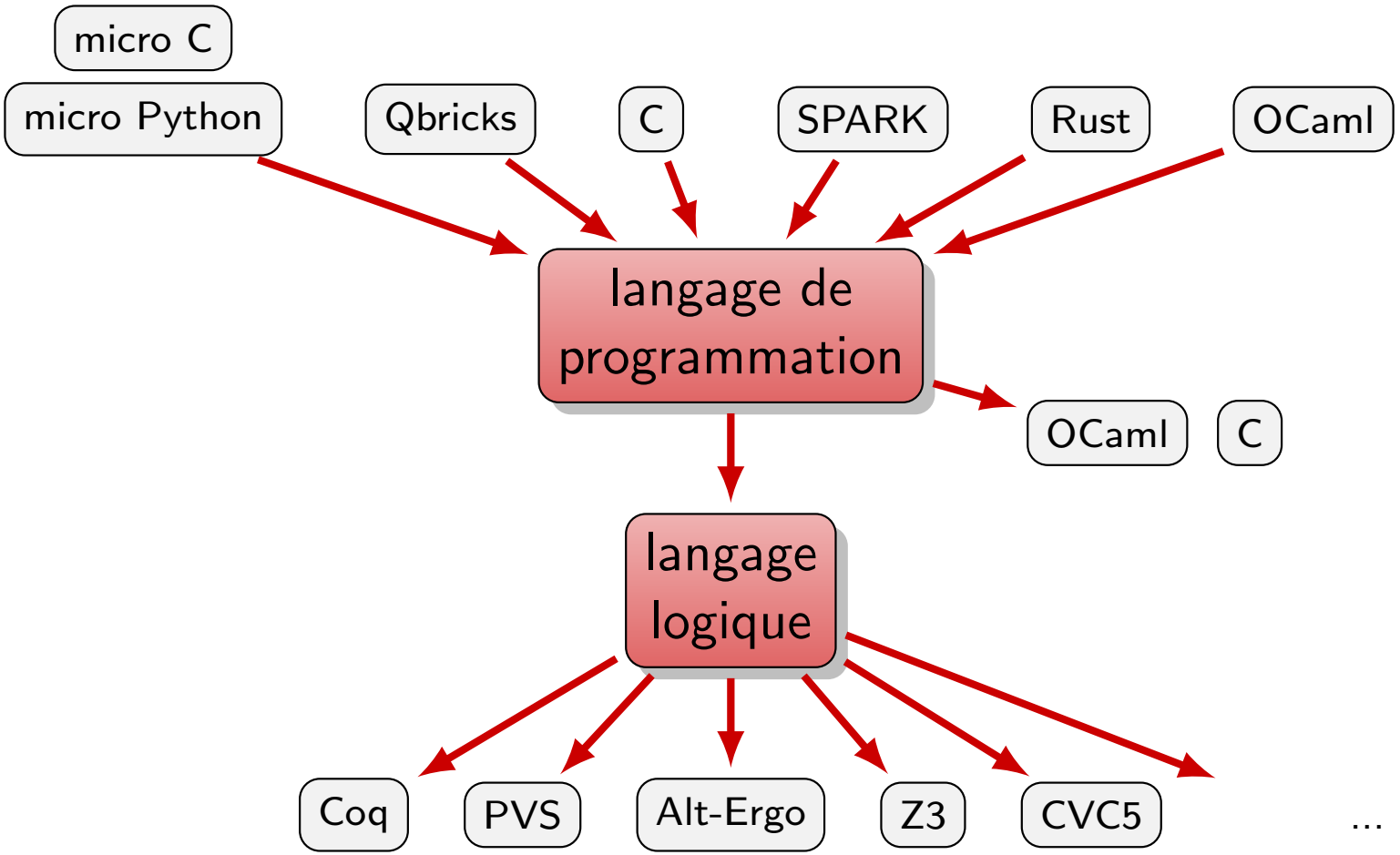
Why3, un couteau suisse



Why3, un couteau suisse



Why3, un couteau suisse



les enjeux de la vérification déductive

écrire une spécification est un processus

- complexe
- coûteux
- qui ne s'automatise pas
- qui comporte des pièges
  -  spécification incomplète
  -  spécification incorrecte

```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

sortie le tableau est trié dans l'ordre croissant

```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

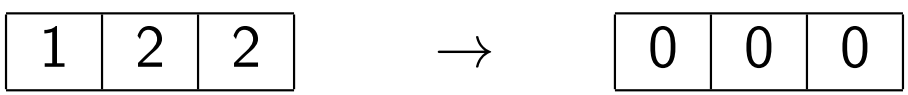
sortie le tableau est trié dans l'ordre croissant

```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant



```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant

tout élément présent en entrée est présent en sortie

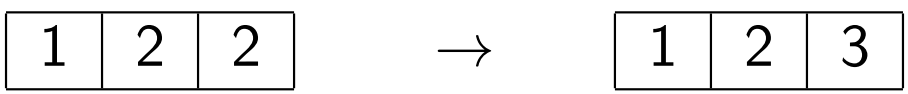
```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant

tout élément présent en entrée est présent en sortie



```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant

tout élément présent en entrée est présent en sortie

tout élément présent en sortie était présent en entrée

```
void yet_another_sort(a: array int)
```

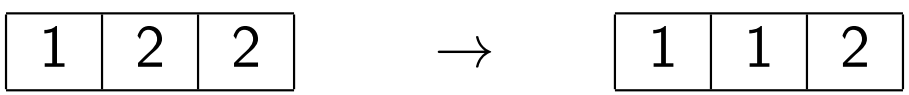
entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant

tout élément présent en entrée est présent en sortie

tout élément présent en sortie était présent en entrée



```
void yet_another_sort(a: array int)
```

entrée un tableau d'entiers

effets le tableau est modifié en place

sortie le tableau est trié dans l'ordre croissant

le tableau est une permutation du tableau initial

```
void yet_another_sort(a: array int)
```

entrée	un tableau d'entiers
effets	le tableau est modifié en place
terminaison	garantie
sortie	le tableau est trié dans l'ordre croissant
	le tableau est une permutation du tableau initial

la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas

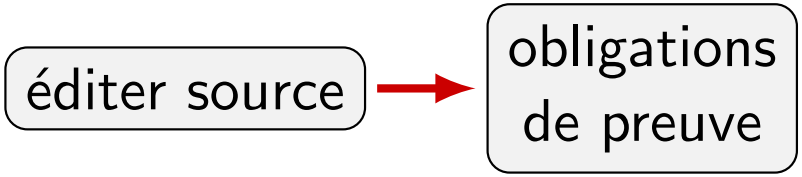
la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas

éditer source

la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas

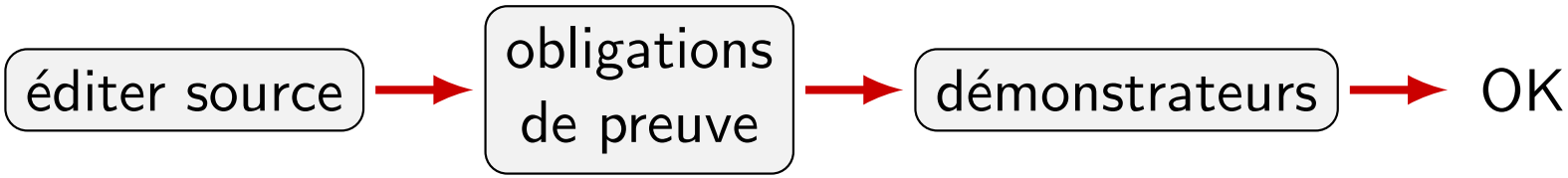


la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas

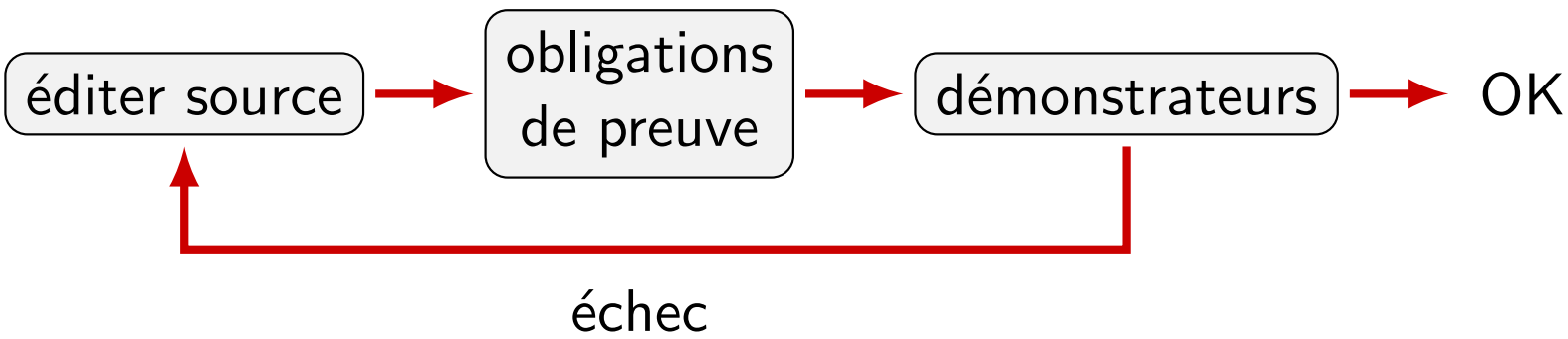
la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas

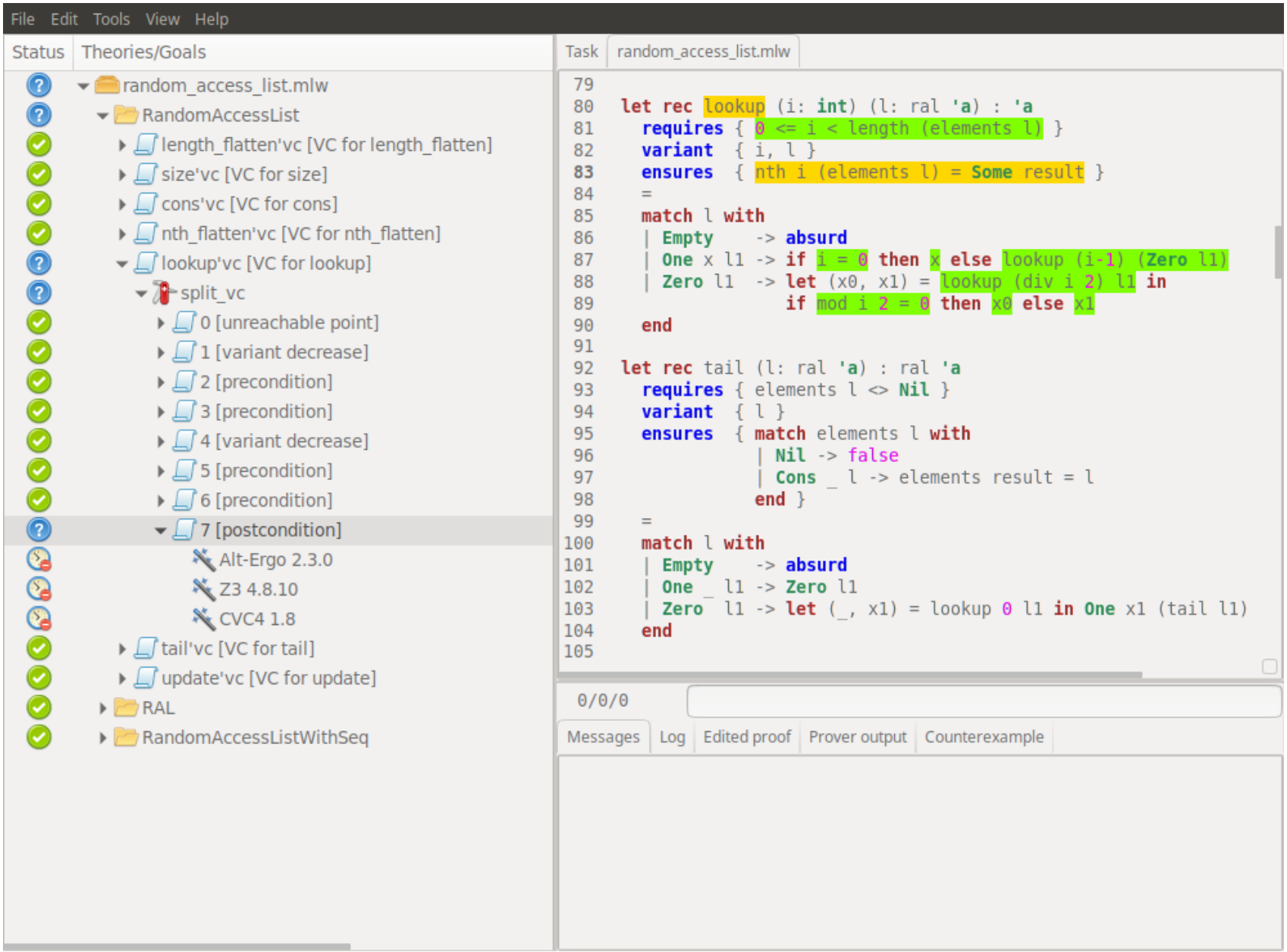


la vérification implique la recherche d'invariants, un processus

- complexe
- coûteux
- qui ne s'automatise pas



Why3 en action



une obligation de preuve non vérifiée peut signifier

- un problème dans la spécification
- un problème dans le code
- un échec du démonstrateur

faire la part des choses demande une grande expertise

leçons d'une expérience avec Parcoursup

travail entamé au LMF en 2019 et aujourd'hui poursuivi au LaBRI

outils

- papier/crayon
- Coq
- JML
- Why3

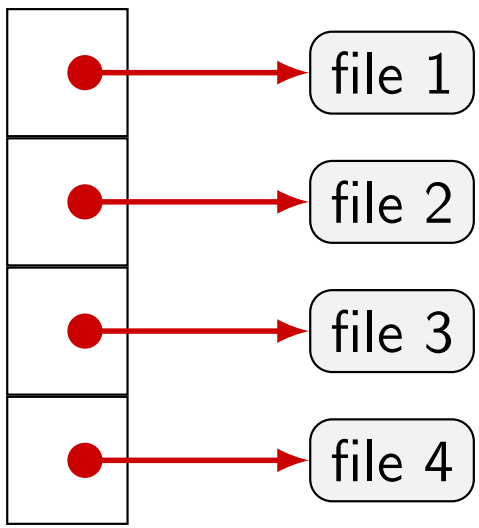
cf *Vérification de l'algorithme de calcul des ordres d'appel dans Parcoursup* (Castéran, Gimbert, Mathieu, Point, 2022)

issue de *Document de présentation des algorithmes de Parcoursup*

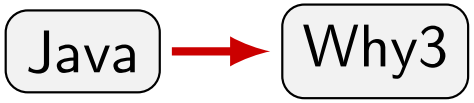
semi-formelle = français + maths

traduite en Why3 / Coq / JML

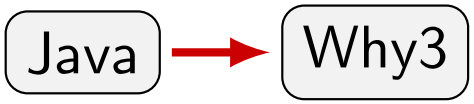
```
EnumMap<TypeCandidat, Queue<VoeuClasse>> filesAttente
```



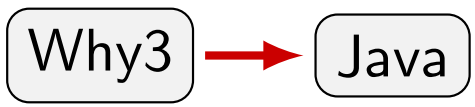
- code déjà écrit



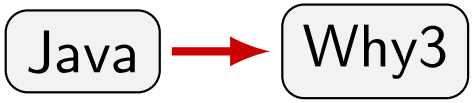
- code déjà écrit



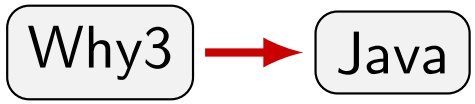
- code produit au final



- code déjà écrit



- code produit au final



- code écrit au fur et à mesure de la preuve
  - possiblement dans un fragment bien identifié (cf SPARK $\subseteq$ Ada)

- la spécification est fondamentale
  - ne s'automatise pas
- la vérification est possible mais coûteuse
  - ne s'automatise pas
- les chercheurs travaillent à réduire l'effort demandé
  - entre science et ingénierie
- on a besoin de développeurs éduqués à la vérification