Preuves, algorithmes et calculs

Fred Mesnard

LIM, université de La Réunion

Octobre 2025

Résumé

Concernant la trilogie *preuves, algorithmes et calculs*, outre le fait que l'éxecution d'un algorithme sur une donnée engendre un calcul, deux autres points de vue se sont dégagés :

- Des preuves, on extrait des algorithmes via la correspondance de Curry-Howard. C'est l'approche mise en œuvre par exemple dans le système Rocq (ex Coq).
- Des preuves, on extrait des calculs. C'est l'approche de la programmation logique.

Nous notons qu'il est également envisageable dans ce dernier cas d'extraire un programme logique. Nous illustrons ces notions par un exemple concret.

Preuves, algorithmes et calculs

Deux points de vue possibles :

- des preuves, on extrait des algorithmes
  - Une correspondance : Curry 58, Howard 69, ...
  - Rocq : Huet & Coquand 84, Paulin 89, ...
  - Le théorème des 4 couleurs, Gonthier Werner 07
  - Le compilateur CompCert, Leroy 08

- des preuves, on extrait des calculs
  - Résolution, Robinson 65
  - Résolution SLD (*Selective Linear Definite* clause resolution), Kowalski & Kuehner 71
  - Prolog, Colmerauer & Roussel 72
  - *Predicate Logic as a Programming Language*, Kowalski 73

Extraction d'algorithmes : un exemple

On se place dans la structure $(\mathbb{N}, \{0, 1\}, \{+/2\}, \{=/2, \leq /2\})$.
Montrons :

$$\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$$

Soit (x, y) un couple d'entiers naturels quelconques :

- Si $y > x$, l'implication est trivialement vraie
- Sinon $z = x - y \in \mathbb{N}$ convient

Mais la soustraction n'existe pas dans la structure considérée $(\mathbb{N}, \{0, 1\}, \{+/2\}, \{=/2, \leq /2\})$!

Extraction d'algorithmes : un exemple

Reprenons en montrant d'abord par induction sur x :

$$\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$$

- Pour x=0, montrons $\forall y [y \leq 0 \rightarrow \exists z | 0 = y + z]$
  Distinguons 2 cas pour y :
  - $y = 0$: $z = 0$ convient
  - $y > 0$: l'implication est trivialement vraie
- Supposons la propriété vraie pour un certain x :
  $HR : \forall y [y \leq x \rightarrow \exists z | x = y + z]$
  Montrons qu'elle reste vraie pour $x' = x + 1$, *i.e.*,
  $\forall y [y \leq x' \rightarrow \exists z | x' = y + z]$.
  Distinguons 3 cas pour y :
  - $y < x + 1$: en appliquant HR, $z' = z + 1$ convient
  - $y = x + 1$: $z = 0$ convient
  - $y > x + 1$: l'implication est trivialement vraie

Extraction d'algorithmes : un exemple

On a donc bien $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$

Comment exprimer z en fonction de x et y ?

En *synthétisant* la soustraction à partir de la démonstration précédente !

Extraction d'algorithmes : un exemple

Rocq peut synthétiser la fonction *minus* qui calcule *z* en fonction de *x* et *y*, correcte par extraction :

```
1  Require Import Lia.
2
3  Lemma minus: ∀ x y : ℕ, y ≤ x → {z | x=y+z}.
4  Proof.
5  intro. induction x.
6  - intros. exists 0. lia.
7  - intros. destruct y.
8    + exists (S x). lia.
9    + destruct (IHx y).
10     * lia.
11     * exists x₀. lia.
12 Defined.
13
14 Require Extraction.
15
16 Extraction minus.
17
18 (** val minus : nat -> nat -> nat **)
19 (*
20 let rec minus x y =
21   match x with
22 | O -> O
23 | S x0 ->
24   (match y with
25   | O -> S x0
26   | S y0 -> minus x0 y0)
27 *)
```

Extraction de calculs par programmation logique

La programmation logique et Prolog :

- La programmation logique est une façon de programmer :
  - on décrit le problème sous forme d'assertions logiques
  - on questionne cette description
  - le système répond
- Prolog est le langage de programmation logique le plus utilisé

Extraction de calculs par programmation logique

Même problème : $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$, cette fois-ci en programmation logique

Représentons les entiers par les termes $\{0, s(0), s(s(0)), ...\}$ et modélisons l'addition :

$$\begin{array}{lll} \forall y & [\top & \rightarrow \quad add(0, y, y)] \\ \forall x\ y\ z & [add(x, y, z) & \rightarrow \quad add(s(x), y, s(z))] \end{array}$$

Idem, syntaxe Prolog :

```
1 add(0,Y,Y).
2 add(s(X),Y,s(Z)) :- add(X,Y,Z).
```

Extraction de calculs par programmation logique

```
1 add(0,Y,Y).
2 add(s(X),Y,s(Z)) :- add(X,Y,Z).
```

- chaque implication est correcte : je lis et je suis ok
  $\Rightarrow$ le programme est correct
  Toute réponse calculée par le système est correcte

La correction des programmes logiques est triviale !

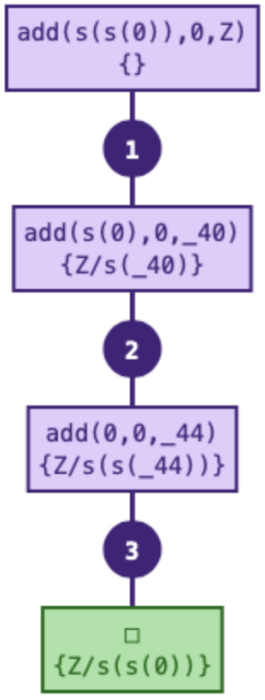
- par un calcul ascendant de la définition inductive :
  $[\ add(x, y, z)\ ] = \{(s^n(0), y, s^n(y)) | n \in \mathbb{N}, y\ terme\}$
  $\Rightarrow$ le programme est largement complet

Pour la complétude, voir Staerk 98, Drabent 22

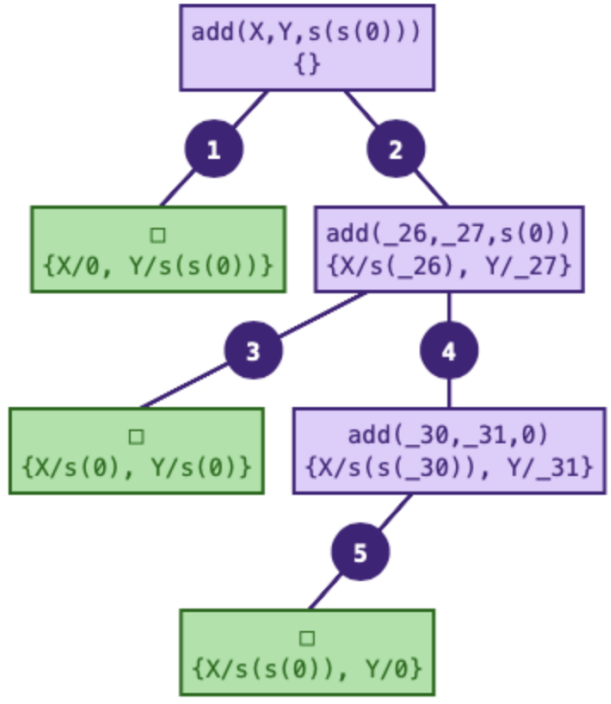
Extraction de calculs par programmation logique

Via la résolution SLD, on extrait les valeurs des preuves :

$\exists ?Z \;\; add(s(s(0)), 0, Z)$



$\exists ?X \;\; Y \;\; add(X, Y, s(s(0)))$



Extraction de calculs par programmation logique

La programmation logique, c'est :

- spécifier en clauses de Horn définies — *quoi*
- prouver procéduralement des requêtes (conjonctions d'atomes existentiellement quantifiés) — *comment*
- extraire les valeurs des variables (toutes $\exists$ quantifiées)

avec comme résultat central (van Emden & Kowalski 76) :

$$M_P = lfp(T_P) = SS(P)$$

qui identifie le point vue logique, point fixe et opérationnel

La programmation logique est une instance du paradigme de programmation *déclarative* : l'utilisateur se concentre sur le *quoi*, la machine gère le *comment*

Extraction de calculs par programmation logique

Retour à notre exemple initial
Dans $\mathbb{N}$, on a : $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$

Définitions Prolog :

```
leq(0,X).
leq(s(Y),s(X)) :- leq(Y,X).

add(0,X,X).
add(s(Y),Z,s(X)) :- add(Y,Z,X).
```

Retour à notre exemple initial

Dans $\mathbb{N}$, on a : $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$

Cette formule ne peut pas se réécrire comme une conjonction existentielle : elle n'est pas directement traitable par Prolog

Mais on *constate* la propriété pour $X = s(s(0))$ et $Y = s(0)$:

```
[?- X=s(s(0)), Y=s(0), leq(Y,X).
X = s(s(0)),
Y = s(0).

[?- X=s(s(0)), Y=s(0), leq(Y,X),add(Y,Z,X).
X = s(s(0)),
Y = Z, Z = s(0).

?-
```

NB : la fonction qui calcule z pour x et y donnés est *implicite* dans la définition *relationnelle* de $add(y, z, x)$

Extraction de calculs par programmation logique

Dans $\mathbb{N}$, on a : $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$

On constate via Prolog la propriété pour $X = s(s(0))$:

```
[?- X = s(s(0)), leq(Y,X).
X = s(s(0)),
Y = 0 ;
X = s(s(0)),
Y = s(0) ;
X = Y, Y = s(s(0)) ;
false.

[?- X = s(s(0)), leq(Y,X), add(Y,Z,X).
X = Z, Z = s(s(0)),
Y = 0 ;
X = s(s(0)),
Y = Z, Z = s(0) ;
X = Y, Y = s(s(0)),
Z = 0 ;
false.
```

Extraction de calculs par programmation logique

Dans $\mathbb{N}$, on a :

$$\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$$

i.e.

$$\forall y \forall x [y \leq x \rightarrow \exists z | x = y + z]$$

On constate via Prolog la propriété pour $Y = s(0)$:

```
[?- Y = s(0), leq(Y,X).
Y = s(0),
X = s(_).

[?- Y = s(0), leq(Y,X), add(Y,Z,X).
Y = s(0),
X = s(Z).

?-
```

Extraction d'algorithmes en programmation logique

Dans $\mathbb{N}$, on a : $\forall x \forall y [y \leq x \rightarrow \exists z | x = y + z]$
Preuve en LPTP :

Lemma 1 $[leq\_add]$ $\forall x, y\, (\mathbf{S}\,\mathtt{leq}(y, x) \rightarrow \exists z\ \mathbf{S}\,\mathtt{add}(y, z, x))$.

Proof.
Induction_0: $\forall y, x\, (\mathbf{S}\,\mathtt{leq}(y, x) \rightarrow \exists z\ \mathbf{S}\,\mathtt{add}(y, z, x))$.
$\quad \text{Hypothesis}_1$: none. $\mathbf{S}\,\mathtt{add}(\mathtt{0}, y, y)$. $\exists z\ \mathbf{S}\,\mathtt{add}(\mathtt{0}, z, y)$.
$\quad \text{Conclusion}_1$: $\exists z\ \mathbf{S}\,\mathtt{add}(\mathtt{0}, z, y)$.
$\quad \text{Hypothesis}_1$: $\exists z\ \mathbf{S}\,\mathtt{add}(x, z, y)$ and $\mathbf{S}\,\mathtt{leq}(x, y)$.
$\quad\quad \text{Let}_2\ z$ with $\mathbf{S}\,\mathtt{add}(x, z, y)$. $\mathbf{S}\,\mathtt{add}(\mathtt{s}(x), z, \mathtt{s}(y))$. $\exists z\ \mathbf{S}\,\mathtt{add}(\mathtt{s}(x), z, \mathtt{s}(y))$.
$\quad\quad \text{Thus}_2$: $\exists z\ \mathbf{S}\,\mathtt{add}(\mathtt{s}(x), z, \mathtt{s}(y))$.
$\quad \text{Conclusion}_1$: $\exists z\ \mathbf{S}\,\mathtt{add}(\mathtt{s}(x), z, \mathtt{s}(y))$. $\square$

Il est possible d'extraire de cette preuve un programme logique qui calcule z en fonction de x et y
Fribourg 90, Bouverot 91 (le système Exexe)

Extraction d'algorithmes en programmation logique

» » » » » » REGLE : init.
» » » » » » BUT : leq(X,Y)=>add(X,?Z,Y).
...
» » » » » » succes total !
clauses d entree-sortie apres depliages:
es(0,Y,Y).
es(s(X),s(Y),Z):-es(X,Y,Z).

Conclusion

Concernant les liens entre les concepts de *preuves, algorithmes et calculs*, nous avons illustré les deux approches classiques. Nous en avons profité pour esquisser deux emplois de la programmation logique.

Le premier est bien connu :

- le mécanisme de la résolution SLD permet d'extraire des preuves les valeurs validant une requête existentiellement quantifiée.

Le second l'est moins :

- la preuve d'une propriété d'un programme logique peut permettre la génération d'un programme logique réalisant cette propriété.

Questions, remarques ?

frederic.mesnard@univ-reunion.fr