Typage pour la Vérification

Matthieu Boyer

Cours TalENS 3 2025-2026

Table des matières

L'objectif de ce cours va être de savoir résoudre le problème suivant :

Problème Mathémagique	
Entrée :	$f_0, \dots, f_{n-1} \in \mathbb{N}^{\mathbb{N}}$
Sortie :	Une preuve que $\mathbb{N}$ est un égaliseur de $f_0 \to \cdots \to f_{n-1}$ et d'une certaine $f : \mathbb{N} \to \{k\}$.

Il s'agit d'une restriction forte (plus précisément, il s'agit d'une restriction aux fonctions entières d'arité 1).

1 Vérification de Programmes

1.1 Programme et Système de Transition

R La définition de cette section n'est pas suffisamment générale pour représenter un vrai langage de programmation, et ne reprend pas que des notations usuelles, mais sert plus à introduire le sujet. Toutefois, elle est suffisamment intéressante pour développer les concepts derrière la vérification de programme.

Définition 1.1 Un système de transition est un couple $(S, \to)$ où :

- S est un ensemble, possiblement infini, d'états ;
- $\to \subseteq S \times S$ est une relation de transition.

On notera $X \to Y$ pour dire que $(X, Y) \in \to$.

Dans notre cas, on va utiliser le système de transition $\mathbb{PN}$ avec $S = \mathcal{P}(\mathbb{N})$ l'ensemble des ensembles de nombres entiers et $X \times Y \in \to$ si et seulement si il existe une fonction (calculable) f telle que $f(X) = Y$. On dit que X est le domaine $\operatorname{dom} f$ de f et Y est son codomaine $\operatorname{codom} f$.

Définition 1.2 Une instruction (fonctionnelle) est une fonction [a] sur $X \subseteq \mathbb{N}$.

Un programme (fonctionnel) est une suite finie d'instructions fonctionnelles, qu'on applique à la suite.

a. Partielle, mais calculable !

Définition 1.3 On notera $f_1 \circ f_0 = f_0; f_1$ la composée de f_0 par f_1, quand le codomaine de f_0 est inclus dans le domaine de f_1. Plus généralement, on notera

$$\bigodot_{i=0}^{k} f_i = f_k \circ \cdots \circ f_0 = f_0; \cdots; f_k.$$

Cependant, considérons le programme suivant :

- $f_0(n \in \mathbb{N}) = 2n + 1$;
- $f_1(n = 2k \in 2\mathbb{N}) = k$.

En langage naturel : à un nombre entier, on le double et on lui ajoute un, puis s'il est pair on le divise par 2. Puisqu'on sait que $2n + 1$ sera toujours un nombre impair, ce programme n'est pas bien défini, puisqu'on ne sait pas quoi faire avec f_1.

Définition 1.4 Un programme $f_0, \ldots, f_{n-1}$ est bien typé si on a

$$\mathbb{N} \xrightarrow{f_0} f_0(\mathbb{N}) \xrightarrow{f_1} \cdots \xrightarrow{f_k} \left(\bigodot_{i=0}^{k} f_i\right)(\mathbb{N}) \xrightarrow{f_{k+1}} \cdots \xrightarrow{f_{n-1}} \left(\bigodot_{i=0}^{n-1} f_i\right)(\mathbb{N}).$$

Ceci signifie seulement que $(f_k)_{|\bigodot_{i \leq k-1} f_i(\mathbb{N})}$ a son codomaine inclus dans le domaine de f_{k+1}.

Théorème 1.5 — Théorème de Rice Il n'existe pas de fonction mathématique calculable qui prend en entrée un programme fonctionnel et qui dit si le programme est bien typé ou non.

1.2 Inférence de Types

Encore une fois, toutes les définitions de cette section sont trop peu générales, mais servent pour introduire des concepts.

Définition 1.6 Un domaine de valeurs/type est un état du système de transition.

Le type d'une fonction, noté $A \to B$ est le plus grand A et plus petit B tel que f est définie par tout sur A et que pour tout $a \in A, f(a) \in B$.

Remarquons alors qu'un programme est bien typé exactement lorsque le type de renvoi de f_i est inclus dans le type de valeurs de f_{i+1}.

On peut inférer le type d'un programme à partir du type des fonctions. Notre objectif de départ devient donc de calculer le type de la fonction composée $f_0; f_1; \cdots; f_{n-1}$ associée au programme.

On introduit pour ça la notion de jugement de typage.

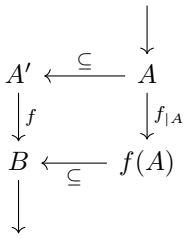
Définition 1.7 Un jugement de typage est une preuve sous forme d'arbre permettant de calculer le type d'une fonction. Elles se créent à partir des blocs de base

$$\overline{\Gamma \vdash f : \mathrm{dom}(f) \to \mathrm{codom}(f)}$$

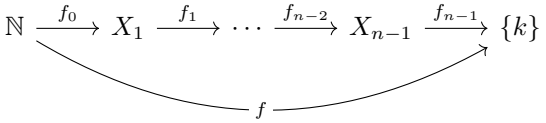
et des règles de composition

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash f : A \to B}{\Gamma \vdash fx : B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash g : B \to C}{\Gamma \vdash f;g : A \to C} \qquad \frac{\Gamma \vdash A \subseteq A' \qquad \Gamma \vdash f : A' \to B}{\Gamma \vdash f(A) \subseteq B}$$

Un programme p est bien typé s'il existe un jugement de typage pour la fonction composée. En pratique, le problème est donc de réduire au minimum le codomaine de f sachant qu'on la restreint à un domaine :



Résoudre le problème de départ revient alors à trouver un certain k tel qu'il existe $f : \mathbb{N} \to \{k\}$ et que le diagramme ci-dessous commute :



En linguistique ?!

En sémantique des langages naturels, on associe à chaque mot une fonction qui représente mathématiquement son sens. On suppose ensuite que la sémantique du langage est compositionnelle, c'est-à-dire que le sens d'une phrase est fonction du sens de ses composantes et de sa structure syntactique.

On attribue un type aux mots en fonction de s'ils représentent une valeur de vérité, une véritable entité, ou une fonction entre des types.

C'est ce qu'on appelle une théorie de sémantique compositionnelle pour le langage naturel.

2 Relations et Quotients

2.1 Exemple de l'arithmétique modulaire

Définition 2.1 La division euclidienne de $a \in \mathbb{Z}$ par $b \in \mathbb{Z}$ est un couple d'entiers (q, r) tel que

- $a = qb + r$;
- $0 \leq r < q$.

On dit que q est le quotient de a par b et r le reste dans la division euclidienne de a par b, ou le reste modulo b de a. On note $r = a \mod b$.

Proposition 2.2 On a, pour a, b, n des entiers

- $a + b \mod n = (a \mod n + b \mod n) \mod n$;
- $a \times b \mod n = (a \mod n \times b \mod n) \mod n$;
- $-a \mod n = -(a \mod n) \mod n$.

Par conséquent, il suffit de calculer avec les restes modulo n pour obtenir le résultat modulo n.

En propageant les calculs modulo n, on va pouvoir restreindre à n le nombre de cas à étudier, et possiblement encore plus le nombre de valeurs de sortie.

2.2 Plus généralement

Définition 2.3 Une relation d'équivalence $\sim$ sur E est une partie de $E \times E$, notée $x \sim y$ et telle que

- $x \sim x$;
- $x \sim y \Leftrightarrow y \sim x$;
- $x \sim y \wedge y \sim z \Rightarrow x \sim z$.

L'égalité est une relation d'équivalence. L'égalité modulo n aussi.

Définition 2.4 La classe d'équivalence de $x \in E$ sous $\sim$ une relation d'équivalence est l'ensemble des éléments de E qui sont en relation avec x.

Le quotient $E/\sim$ de E par $\sim$ une relation d'équivalence sur E est l'ensemble des classes d'équivalence des éléments de E sous $\sim$.

Définition 2.5 On dit qu'une opération $f : E \to E$ passe au quotient si le diagramme ci-dessous commute.

$$\begin{array}{ccc} E & \xrightarrow{/\sim} & E/\sim \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f} \\ E & \xrightarrow[/\sim]{} & E/\sim \end{array}$$

Proposition 2.6 Une fonction $f : E \to X$ définit une relation d'équivalence $\mathcal{R}$ par $x\mathcal{R}y$ si et seulement si $f(x) = f(y)$.