

Bootstrap

Erik Helmers

May 31, 2023

Contents

1	Design	2
1.1	Préliminaires : Types dépendants	2
1.2	Description des structures inductives	3
2	Construction	3
2.1	Lambda calcul minimal	3
2.1.1	Syntaxe	4
2.1.2	Contexte	4
2.1.3	Evaluation	4
2.1.4	Typage	5
2.2	Interlude : booleans	6
2.2.1	Syntaxe	7
2.2.2	Evaluation	7
2.2.3	Typing	7
2.3	Enumerations	8
2.3.1	Syntax	8
2.3.2	Evaluation	8
2.3.3	Typing	9
2.4	Descriptions	10
2.4.1	Syntax	10
2.4.2	Evaluation	10
2.4.3	Typing	11

L'objectif de ce projet de recherche est d'implémenter un système de type qui intègre la descriptions des types au sein de la théorie des types.

1 Design

1.1 Préliminaires : Types dépendants

Les types dépendants constituent le socle de ce projet, il est donc de bon ton de commencer par un rapide rappel sur la notion.

Le lecteur familier avec un langage de programmation typé moderne aura été exposé à la notion de type générique (ou polymorphique). Ce sont les types qui sont fonctions d'autres types. Par exemple, les listes en OCaml sont décrites par `'a list` où `'a` représente nécessairement un type (`int`, `char`, ...). Ce mécanisme rend plus ergonomique la manipulation de structure de données.

Les types dépendants eux, sont fonction de termes. Un exemple habituel est celui des listes dont la longueur est représentée dans le type. On pourrait imaginer la notation `(int * 'a) list` où `(5, int) list` est le type des listes d'entier de longueur 5.

Ce qui rend cruciale l'introduction de cette expressivité, est qu'elle permet d'encoder, en plus de sa structure, la logique d'un type. Par nature, les types dépendants décrivent donc un comportement dynamique.

Pour illustrer ce propos, prenons en exemple les fonctions de choix de l'axiome éponyme.

- Mathématiquement, soit X un ensemble, alors une fonction f est de choix si elle a pour structure

$$f : \mathcal{P}(X) \rightarrow X$$

et qu'elle vérifie

$$\forall S \in \mathcal{P}(X), f(S) \in S$$

- Dans un langage simplement typé, en supposant que X et $\mathcal{P}(X)$ désignent des types, on peut décrire la structure d'une fonction de choix:

```
val f : P(X) -> X
```

- Avec des types dépendants, il est aussi possible d'en décrire la logique:

```
val f : (S : P(X)) -> S
```

Un lecteur attentif pourrait remarquer que cette déclaration n'est pas strictement compatible, à priori, avec les règles données en ???. Cela n'entre pas dans le périmètre de ce projet, mais il est possible de résoudre ce problème en introduisant une hiérarchie des types.

De l'introduction des types dépendants, il découle un phénomène supplémentaire. Types et valeurs sont nécessairement aplatis sur le même plan syntaxique. Le gain d'expressivité est donc aussi accompagné d'une simplification du langage. L'objectif final de ce projet est de reproduire un résultat analogue, cette fois sur la description de types dépendants inductifs.

1.2 Description des structures inductives

Nous allons commencer par faire écho à l'introduction de la section précédente.

```
type 'a list = Nil | Cons of 'a * 'a list

type (n: int, 'a) list = match n with
| Zero -> Nil
| Suc n' -> Cons of 'a * (n', 'a) list
```

La description des types ne fait pas parti du langage mais d'un métalangage. De ce fait, il est impossible de les manipuler, ni même de les inspecter dans notre théorie des types.

2 Construction

Cette partie est la construction progressive d'un système de type suffisant pour décrire des types en son sein.

2.1 Lambda calcul minimal

La première étape du projet consiste en l'établissement d'un interpréteur avec type dépendant.

2.1.1 Syntaxe

$e, \sigma, \kappa ::=$	$e : \sigma$	annotated term
	x	variable
	$\lambda x \mapsto e$	lambda
	$e \ e'$	application
	$(x : \sigma) \rightarrow \sigma'$	pi type
	(e, e')	tuple
	$\text{fst } e$	fst
	$\text{snd } e$	snd
	$(x : \sigma) \times \sigma'$	sigma type
	\star	type of types

où e, σ, κ représentent des valeurs, des types et des kinds respectivement.

2.1.2 Contexte

$\Gamma ::=$	ϵ	empty context
	$\Gamma, x : \tau$	adding a variable

$$\frac{}{\text{valid}(\epsilon)} \quad \frac{\text{valid}(\Gamma) \quad \Gamma \vdash \tau \Leftarrow \star}{\text{valid}(\Gamma, x : \tau)}$$

2.1.3 Evaluation

$\nu, \tau ::=$	n	neutral term
	$\lambda x \mapsto \nu$	lambda
	$(x : \tau) \rightarrow \tau'$	dependent function space
	(ν, ν')	tuple
	$(x : \tau) \times \tau'$	dependent pair space
	\star	type of types
$n ::=$	x	variable
	$n \ \nu$	neutral app
	$\text{fst } n$	neutral first projection
	$\text{snd } n$	neutral second projection

$$\begin{array}{c}
\frac{}{\star \Downarrow \star} \text{ (STAR)} \qquad \frac{}{x \Downarrow x} \text{ (VAR)} \qquad \frac{e \Downarrow \nu}{e : \sigma \Downarrow \nu} \text{ (ANN)} \\
\\
\frac{e \Downarrow \nu}{\lambda x \mapsto e \Downarrow \lambda x \mapsto \nu} \text{ (LAM)} \qquad \frac{e \Downarrow \nu \quad e' \Downarrow \nu'}{(e, e')(\Downarrow, \nu)\nu'} \text{ (TUPLE)} \\
\\
\frac{e \Downarrow \lambda x \mapsto \nu \quad \nu[x \mapsto e'] \Downarrow \nu'}{e \ e' \Downarrow \nu'} \text{ (APP)} \qquad \frac{e \Downarrow n \quad e' \Downarrow \nu'}{e \ e' \Downarrow n \ \nu'} \text{ (NAPP)} \\
\\
\frac{e \Downarrow (\nu, \nu')}{\text{fst } e \Downarrow \nu} \text{ (FST)} \qquad \frac{e \Downarrow (\nu, \nu')}{\text{snd } e \Downarrow \nu'} \text{ (SND)} \qquad \frac{e \Downarrow n}{\text{fst } e \Downarrow \text{fst } n} \text{ (NFST)} \\
\\
\frac{e \Downarrow n}{\text{snd } e \Downarrow \text{snd } n} \text{ (NSND)} \qquad \frac{\sigma \Downarrow \tau \quad \sigma' \Downarrow \tau'}{(x : \sigma) \rightarrow \sigma' \Downarrow (x : \tau) \rightarrow \tau'} \text{ (PI)} \\
\\
\frac{\sigma \Downarrow \tau \quad \sigma' \Downarrow \tau'}{(x : \sigma) \times \sigma' \Downarrow (x : \tau) \times \tau'} \text{ (SIGMA)}
\end{array}$$

2.1.4 Typage

Dans la suite, nous notons $e \Rightarrow \tau$ quand le terme e se synthétise en τ et $e \Leftarrow \tau$ quand il est possible de vérifier que e est un τ .

$$\frac{\Gamma \vdash e \Rightarrow \tau}{\Gamma \vdash e \Leftarrow \tau} \text{ (CHK)} \qquad \frac{\Gamma \vdash \sigma \Leftarrow \star \quad \sigma \Downarrow \tau \quad \Gamma \vdash e \Leftarrow \tau}{\Gamma \vdash (e : \sigma) \Rightarrow \tau} \text{ (ANN)}$$

$$\frac{}{\Gamma \vdash \star \Leftarrow \star} \text{ (STAR)} \qquad \frac{\Gamma(x) = \tau}{\Gamma \vdash x \Rightarrow \tau} \text{ (VAR)}$$

$$\frac{\Gamma, x : \tau \vdash e \Leftarrow \tau'}{\Gamma \vdash \lambda x \mapsto e \Leftarrow (x : \tau) \rightarrow \tau'} \text{ (LAM)}$$

$$\frac{\Gamma \vdash e \Leftarrow \tau \quad \Gamma \vdash e' \Leftarrow \tau'}{\Gamma \vdash (e, e') \Leftarrow (x : \tau) \times \tau'} \text{ (TUPLE)}$$

$$\frac{\Gamma \vdash e \Rightarrow (x : \tau) \rightarrow \tau' \quad \Gamma \vdash e' \Leftarrow \tau \quad \tau'[x \mapsto e'] \Downarrow \tau''}{\Gamma \vdash e e' \Rightarrow \tau''} \text{ (APP)}$$

$$\frac{\Gamma \vdash e \Rightarrow (x : \tau) \times \tau'}{\Gamma \vdash \text{fst } e \Rightarrow \tau} \text{ (FST)}$$

$$\frac{\Gamma \vdash e \Rightarrow (x : \tau) \times \tau' \quad \tau'[x \mapsto \text{fst } e] \Downarrow \tau''}{\Gamma \vdash \text{snd } e \Rightarrow \tau''} \text{ (SND)}$$

$$\frac{\Gamma \vdash \sigma \Leftarrow \star \quad \sigma \Downarrow \tau \quad \Gamma, x : \tau \vdash \sigma' \Leftarrow \star}{\Gamma \vdash (x : \sigma) \rightarrow \sigma' \Leftarrow \star} \text{ (PI)}$$

$$\frac{\Gamma \vdash \sigma \Leftarrow \star \quad \sigma \Downarrow \tau \quad \Gamma, x : \tau \vdash \sigma' \Leftarrow \star}{\Gamma \vdash (x : \sigma) \times \sigma' \Leftarrow \star} \text{ (SIGMA)}$$

2.2 Interlude : booleans

At this point, it is useful to be able to run some sanity checks on the implementation. This is removed later, once we get our descriptions working.

2.2.1 Syntaxe

$$\begin{array}{lcl}
 e, \sigma, \kappa & ::= & \dots \\
 & | & \text{true} \\
 & | & \text{false} \\
 & | & \text{cond } e \ [x.\sigma] \ e' \ e'' \quad \text{condition} \\
 & | & \text{bool} \quad \text{type of a bool}
 \end{array}$$

2.2.2 Evaluation

$$\begin{array}{lcl}
 \nu, \tau & ::= & \dots \\
 & | & \text{true} \\
 & | & \text{false} \\
 & | & \text{bool} \\
 \\
 n & ::= & \dots \\
 & | & \text{cond } \nu \ [x.\tau] \ \nu' \ \nu''
 \end{array}$$

$$\frac{}{\text{true} \Downarrow \text{true}} \text{ (TRUE)} \qquad \frac{}{\text{false} \Downarrow \text{false}} \text{ (FALSE)}$$

$$\frac{e \Downarrow \text{true} \quad e' \Downarrow \nu}{\text{cond } e \ [x.B] \ e' \ e'' \Downarrow \nu} \text{ (COND T)} \qquad \frac{e \Downarrow \text{true} \quad e'' \Downarrow \nu}{\text{cond } e \ [x.B] \ e' \ e'' \Downarrow \nu} \text{ (COND F)}$$

$$\frac{e \Downarrow n \quad e' \Downarrow \nu \quad e'' \Downarrow \nu'}{\text{cond } e \ [x.B] \ e' \ e'' \Downarrow \text{cond } n \ [x.\tau] \ \nu \ \nu'} \text{ (NCOND)}$$

$$\frac{}{\text{bool} \Downarrow \text{bool}} \text{ (BOOLT Y)}$$

2.2.3 Typing

$$\frac{}{\text{true} \Leftarrow \text{bool}} \text{ (TRUE)} \qquad \frac{}{\text{false} \Leftarrow \text{bool}} \text{ (FALSE)}$$

$$\frac{\Gamma \vdash e \Leftarrow \text{bool} \quad \Gamma, x : \text{bool} \vdash B \Leftarrow \star \quad B[x \mapsto e] \Downarrow \tau}{\Gamma \vdash \text{cond } e \ [x.B] \ e' \ e'' \Rightarrow \tau} \text{ (COND)}$$

$$\frac{}{\text{bool} \Leftarrow \star} \text{ (BOOLT Y)}$$

2.3 Enumerations

2.3.1 Syntax

e, σ, κ	$::=$...
		nil
		unit
		't label
		label label type
		nil
		[t l]
		labels labels type

2.3.2 Evaluation

$$\frac{l \Downarrow \text{nil}}{\text{record } l \text{ as } x \text{ return } B \Downarrow \text{unit}} \text{ (RECORDNIL)}$$

$$\frac{B[x \mapsto 0] \Downarrow \tau \quad \frac{l \Downarrow [t \ l']}{\text{record } l' \text{ as } x \text{ return } B[x \mapsto 1 + x] \Downarrow \tau'} \text{ (RECORDCONS)}}{\text{record } l \text{ as } x \text{ return } B \Downarrow \tau \times \tau'}$$

$$\frac{l \Downarrow n}{\text{record } l \text{ as } x \text{ return } B \Downarrow \text{record } n \text{ as } x \text{ return } B} \text{ (NRECORD)}$$

$$\frac{e \Downarrow 0 \quad \text{fst } cs \Downarrow \nu}{\text{case } e \text{ as } x \text{ return } B \text{ with } cs \Downarrow \nu} \text{ (CASEZE)}$$

$$\frac{\text{snd } cs \Downarrow cs' \quad \frac{e \Downarrow 1 + e'}{\text{case } e' \text{ as } x \text{ return } B[x \mapsto 1 + x] \text{ with } cs' \Downarrow \nu}}{\text{case } e \text{ as } x \text{ return } B \text{ with } cs \Downarrow \nu} \text{ (CASESUC)}$$

$$\frac{e \Downarrow n \quad cs \Downarrow \nu}{\text{case } e \text{ as } x \text{ return } B \text{ with } cs \Downarrow \text{case } n \text{ as } x \text{ return } B \text{ with } \nu} \text{ (CASEZE)}$$

2.3.3 Typing

$$\begin{array}{c}
\frac{}{\text{nil} \Leftarrow \text{unit}} \text{ (NIL)} \qquad \frac{}{\text{unit} \Leftarrow \star} \text{ (UNIT)} \\
\\
\frac{}{t \Leftarrow \text{label}} \text{ (LABEL)} \qquad \frac{}{\text{label} \Leftarrow \star} \text{ (LABELTY)} \\
\\
\frac{}{\text{nil} \Leftarrow \text{labels}} \text{ (NILL)} \qquad \frac{\Gamma \vdash t \Leftarrow \text{label} \quad \Gamma \vdash l \Leftarrow \text{labels}}{[t \ l] \Leftarrow \text{labels}} \text{ (CONSL)} \\
\\
\frac{}{\text{labels} \Leftarrow \star} \text{ (LABELSTY)} \\
\\
\frac{\Gamma \vdash t \Leftarrow \text{label} \quad \Gamma \vdash l \Leftarrow \text{labels}}{\Gamma \vdash 0 \Leftarrow \text{enum } [t \ l]} \text{ (ZERO)} \\
\\
\frac{\Gamma \vdash t \Leftarrow \text{label} \quad \Gamma \vdash l \Leftarrow \text{labels} \quad \Gamma \vdash n \Leftarrow \text{enum } l}{\Gamma \vdash 1 + n \Leftarrow \text{enum } [t \ l]} \text{ (SUC)} \\
\\
\frac{\Gamma \vdash l \Leftarrow \text{labels}}{\Gamma \vdash \text{enum } l \Leftarrow \star} \text{ (ENUM)} \\
\\
\frac{\Gamma \vdash l \Leftarrow \text{labels} \quad \Gamma, x : \text{enum } l \vdash B \Leftarrow \star}{\Gamma \vdash \text{record } l \text{ as } x \text{ return } B \Rightarrow \star} \text{ (RECORD)} \\
\\
\frac{\Gamma \vdash e \Leftarrow \text{enum } l \quad \Gamma, x : \text{enum } l \vdash B \Leftarrow \star \quad B[x \mapsto e] \Downarrow \tau \quad \Gamma \vdash \text{cs} \Leftarrow \text{record } l \text{ as } x \text{ return } B}{\Gamma \vdash \text{case } e \text{ as } x \text{ return } B \text{ with cs} \Rightarrow \tau} \text{ (CASE)}
\end{array}$$

2.4 Descriptions

2.4.1 Syntax

e, σ	$::=$...	
		'unit	
		'var	identity functor
		' Σ σ e	
		' Π σ e	
		$\llbracket e \rrbracket$ σ	
		desc	descriptor type
		μ e	
		ctor e	

2.4.2 Evaluation

ν, τ	$::=$...	
		'unit	
		'var	identity functor
		' Σ τ D	
		' Π τ D	
		desc	descriptor type
		μ	fixpoint
		ctor n	constructor
n	$::=$...	
		$\llbracket n \rrbracket$ τ	

$$\frac{D \Downarrow \text{'unit}}{\llbracket D \rrbracket \sigma \Downarrow \text{unit}} \text{ (DECODENIL)} \qquad \frac{D \Downarrow \text{'var} \quad \sigma \Downarrow \tau}{\llbracket D \rrbracket \sigma \Downarrow \tau} \text{ (DECODEVAR)}$$

$$\frac{D \Downarrow \text{' Σ τ D' } \quad (e : \tau) \times \llbracket D' e \rrbracket \sigma \Downarrow \tau'}{\llbracket D \rrbracket \sigma \Downarrow \tau'} \text{ (DECODESIGMA)}$$

$$\frac{D \Downarrow \text{' Π τ D' } \quad (e : \tau) \rightarrow \llbracket D' e \rrbracket \sigma \Downarrow \tau'}{\llbracket D \rrbracket \sigma \Downarrow \tau'} \text{ (DECODEPI)}$$

2.4.3 Typing

$$\begin{array}{c}
\overline{\Gamma \vdash \text{'unit'} \Leftarrow \text{desc}} \qquad \overline{\Gamma \vdash \text{'var'} \Leftarrow \text{desc}} \qquad \overline{\Gamma \vdash \text{desc} \Leftarrow \star} \\
\\
\frac{\Gamma \vdash \sigma \Leftarrow \star \quad \sigma \Downarrow \tau \quad \Gamma \vdash D \Leftarrow \tau \rightarrow \text{desc}}{\Gamma \vdash \text{'}\Sigma \sigma D \Leftarrow \text{desc}} \text{(DSIGMA)} \\
\\
\frac{\Gamma \vdash \sigma \Leftarrow \star \quad \sigma \Downarrow \tau \quad \Gamma \vdash D \Leftarrow \tau \rightarrow \text{desc}}{\Gamma \vdash \text{'}\Pi \sigma D \Leftarrow \text{desc}} \text{(DPI)} \\
\\
\frac{\Gamma \vdash D \Leftarrow \text{desc} \quad \Gamma \vdash \sigma \Leftarrow \star}{\Gamma \vdash \llbracket D \rrbracket \sigma \Rightarrow \star} \text{(DECODE)} \qquad \frac{\Gamma \vdash D \Leftarrow \text{desc}}{\Gamma \vdash \mu D \Leftarrow \star} \text{(MU)} \\
\\
\frac{\Gamma \vdash \sigma \Leftarrow \llbracket D \rrbracket (\mu D)}{\Gamma \vdash \text{ctor } \sigma \Leftarrow \mu D}
\end{array}$$