Effectfully gardening with the Pythia

Continuity in a dependent setting

Martin Baillon

Chocola

20th October 2022

Gallinette team, Inria Rennes-Bretagne-Atlantique

Introduction

Theorem

Any function $\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is continuous

Introduction

Theorem

Any function $\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is continuous

What is BTT ?

What does it mean to be continuous ?

How do we prove it ?

Introduction

At the end of this talk, you will know :

- what continuity is, and why it is linked to effects
- why it is difficult to mix MLTT (Coq) with effects
- how BTT solves some problems (but not all)
- how to prove continuity for BTT

I. Continuity

Simple example

$$
\begin{aligned}
\mathtt{f} \quad &: \quad \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\
\mathtt{f}\ \alpha \quad &:= \quad 2 \times (\alpha\ (1 + (\alpha\ 0)))
\end{aligned}
$$

I. Continuity

Simple example

$$\begin{aligned} \texttt{f} &: \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \texttt{f}\ \alpha &:= 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\alpha := \lambda(n : \mathbb{N}).\ n$$

I. Continuity

Simple example

$$\texttt{f} : \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N}$$

$$\texttt{f}\ \alpha := 2 \times (\alpha\ (1 + (\alpha\ 0)))$$

$$\alpha := \lambda(n : \mathbb{N}).\ n$$

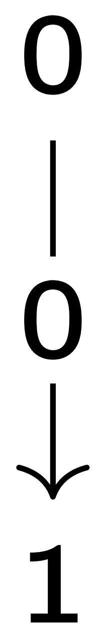
0

I. Continuity

Simple example

$$\begin{aligned} \mathtt{f} \quad &: \quad \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \mathtt{f}\ \alpha \quad &:= \quad 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\alpha \quad := \quad \lambda(n : \mathbb{N}).\ n$$



0

0

1

I. Continuity

Simple example

$$\begin{aligned} \mathtt{f} &: \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \mathtt{f}\ \alpha &:= 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\alpha := \lambda(n : \mathbb{N}).\ n$$

$$\begin{tikzcd} 0 \arrow[d, dash] \\ 0 \arrow[d] \\ 1 \arrow[d, dash] \\ 1 \arrow[d] \\ 2 \end{tikzcd}$$

I. Continuity

Simple example

$$\begin{aligned} \mathtt{f} \quad &: \quad \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \mathtt{f}\ \alpha \quad &:= \quad 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\begin{aligned} \alpha \quad &:= \quad \lambda(n : \mathbb{N}).\ n \\ \beta\ 0 \quad &\equiv \quad 0 \\ \beta\ 1 \quad &\equiv \quad 1 \\ \beta\ n \quad &\equiv \quad \text{underspecified} \end{aligned}$$

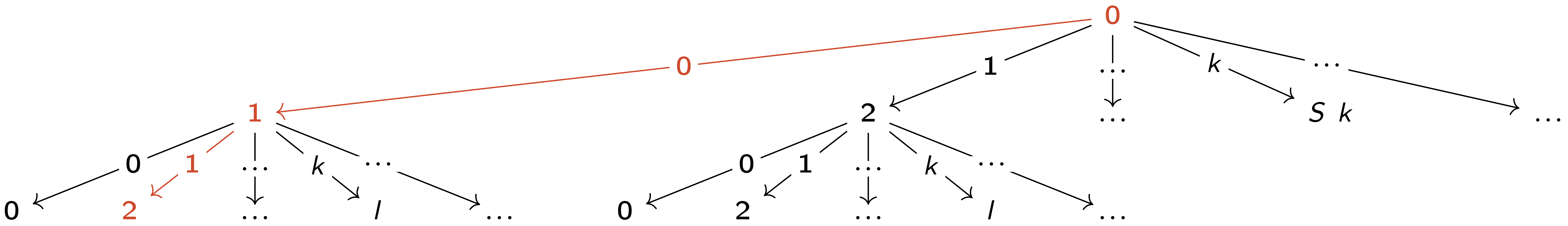
$$\begin{tikzcd} 0 \arrow[d, dash] \\ 0 \arrow[d] \\ 1 \arrow[d, dash] \\ 1 \arrow[d] \\ 2 \end{tikzcd}$$

I. Continuity

Simple example

$$\begin{aligned} \mathtt{f} \quad &: \quad \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \mathtt{f}\ \alpha \quad &:= \quad 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\begin{aligned} \alpha \quad &:= \quad \lambda(n : \mathbb{N}).\ n \\ \beta\ 0 \quad &\equiv \quad 0 \\ \beta\ 1 \quad &\equiv \quad 1 \\ \beta\ n \quad &\equiv \quad \text{underspecified} \end{aligned}$$

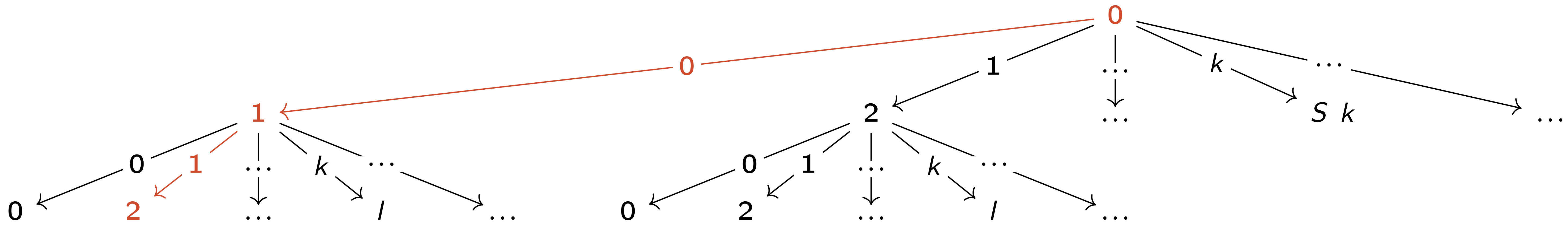


I. Continuity

Simple example

$$\begin{aligned} \mathtt{f} &: \Pi(\alpha : \mathbb{N} \to \mathbb{N}).\, \mathbb{N} \\ \mathtt{f}\ \alpha &:= 2 \times (\alpha\ (1 + (\alpha\ 0))) \end{aligned}$$

$$\begin{aligned} \alpha &:= \lambda(n : \mathbb{N}).\ n \\ \beta\ 0 &\equiv 0 \\ \beta\ 1 &\equiv 1 \\ \beta\ n &\equiv \text{underspecified} \end{aligned}$$



A function $\mathtt{f} : (\mathbb{N} \to \mathbb{N}) \to A$ is said **continuous** if there exists such a tree.

I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

$$
\begin{array}{l}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \quad := \\
\quad | \quad \eta : A \to \mathfrak{D}\ A \\
\quad | \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.
\end{array}
$$

I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

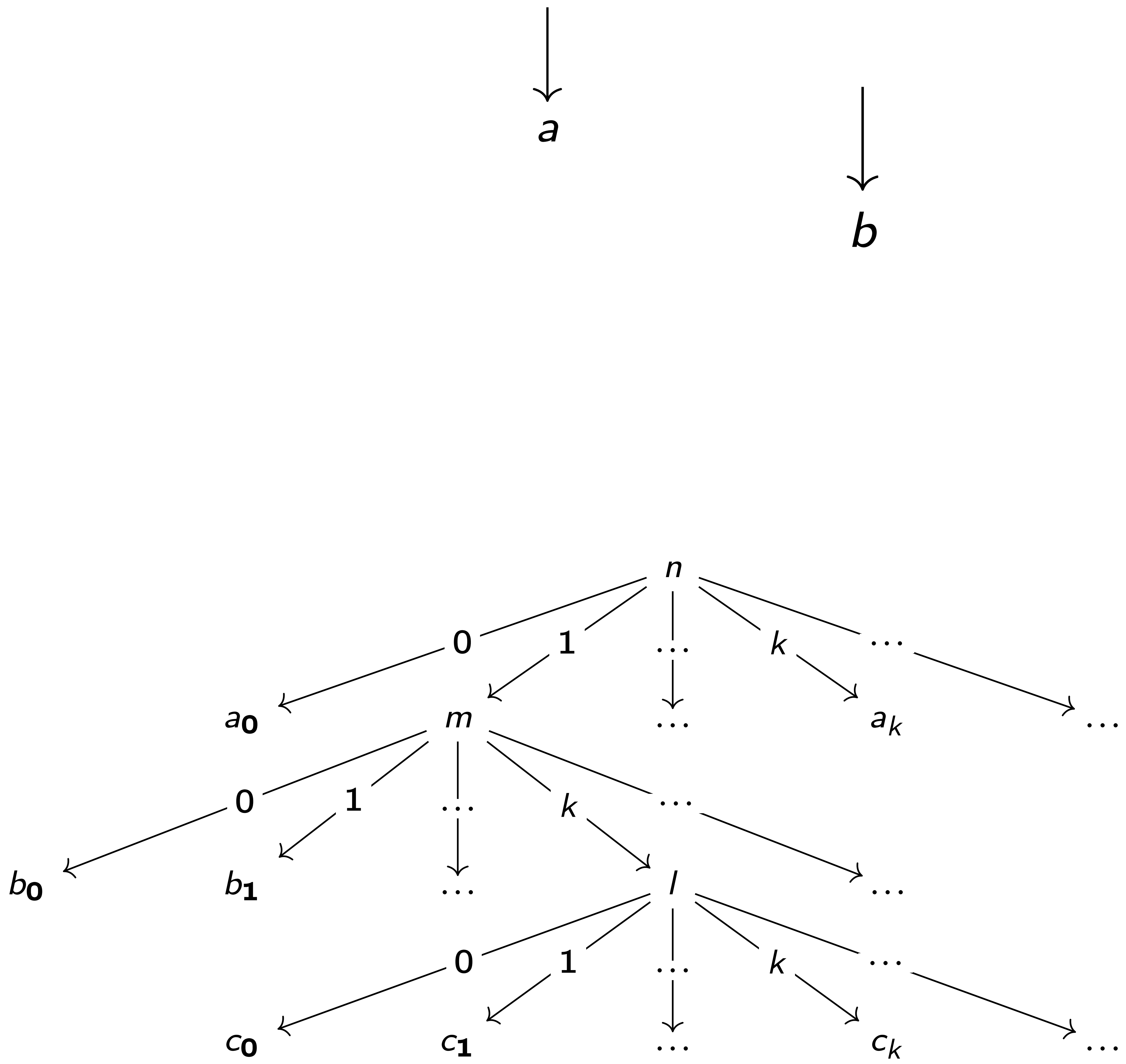
$$
\begin{array}{ll}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\
\mid \quad \eta : A \to \mathfrak{D}\ A & \\
\mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. &
\end{array}
$$

I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

$$\begin{aligned}&\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \quad := \\ &\quad |\quad \eta : A \to \mathfrak{D}\ A \\ &\quad |\quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.\end{aligned}$$



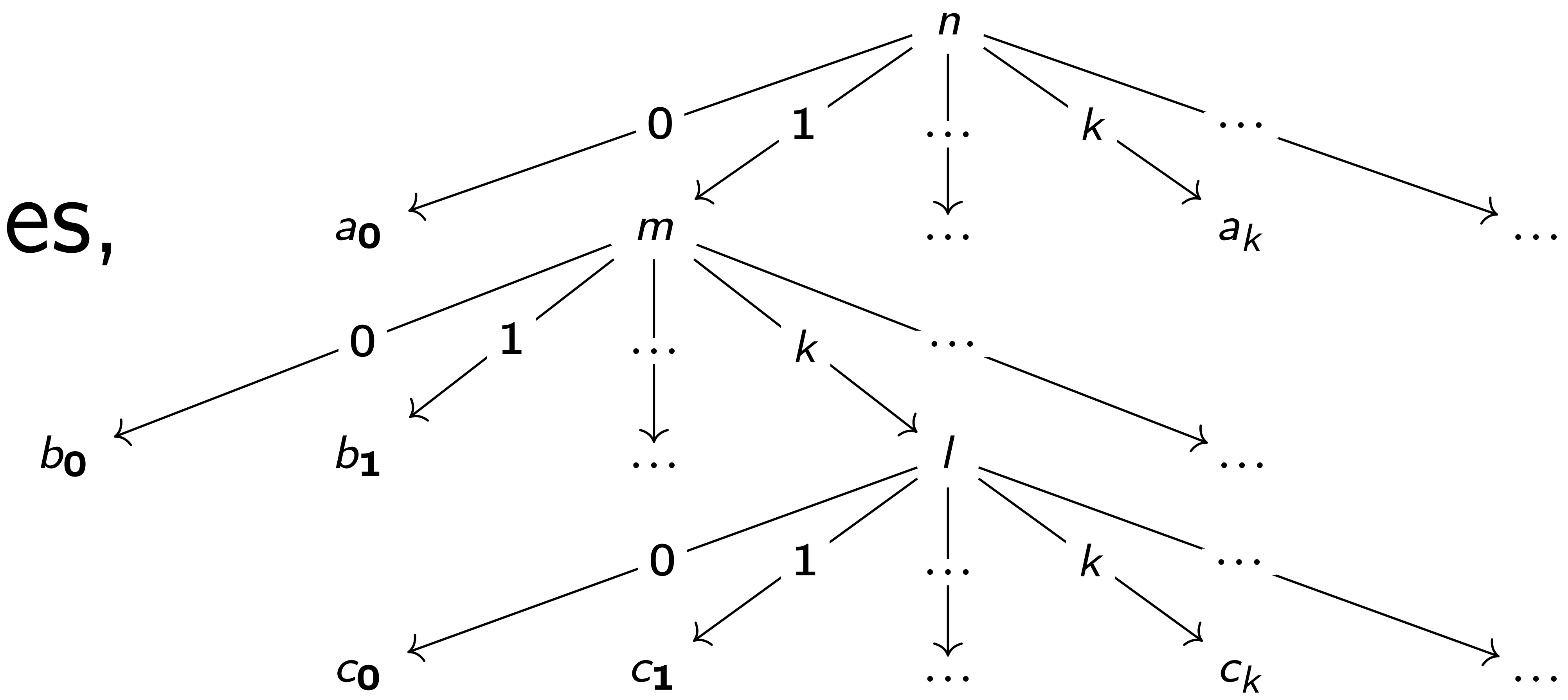
I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

$$\begin{aligned}&\texttt{Inductive}\ \mathfrak{D}\ (A:\square):\square \qquad := \\ &\quad|\qquad \eta : A \to \mathfrak{D}\ A \\ &\quad|\qquad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.\end{aligned}$$

$\mathfrak{D}\ A$ is the type of well-founded, $\mathbb{N}$-branching trees, with inner nodes labeled in $\mathbb{N}$ and leaves in A.

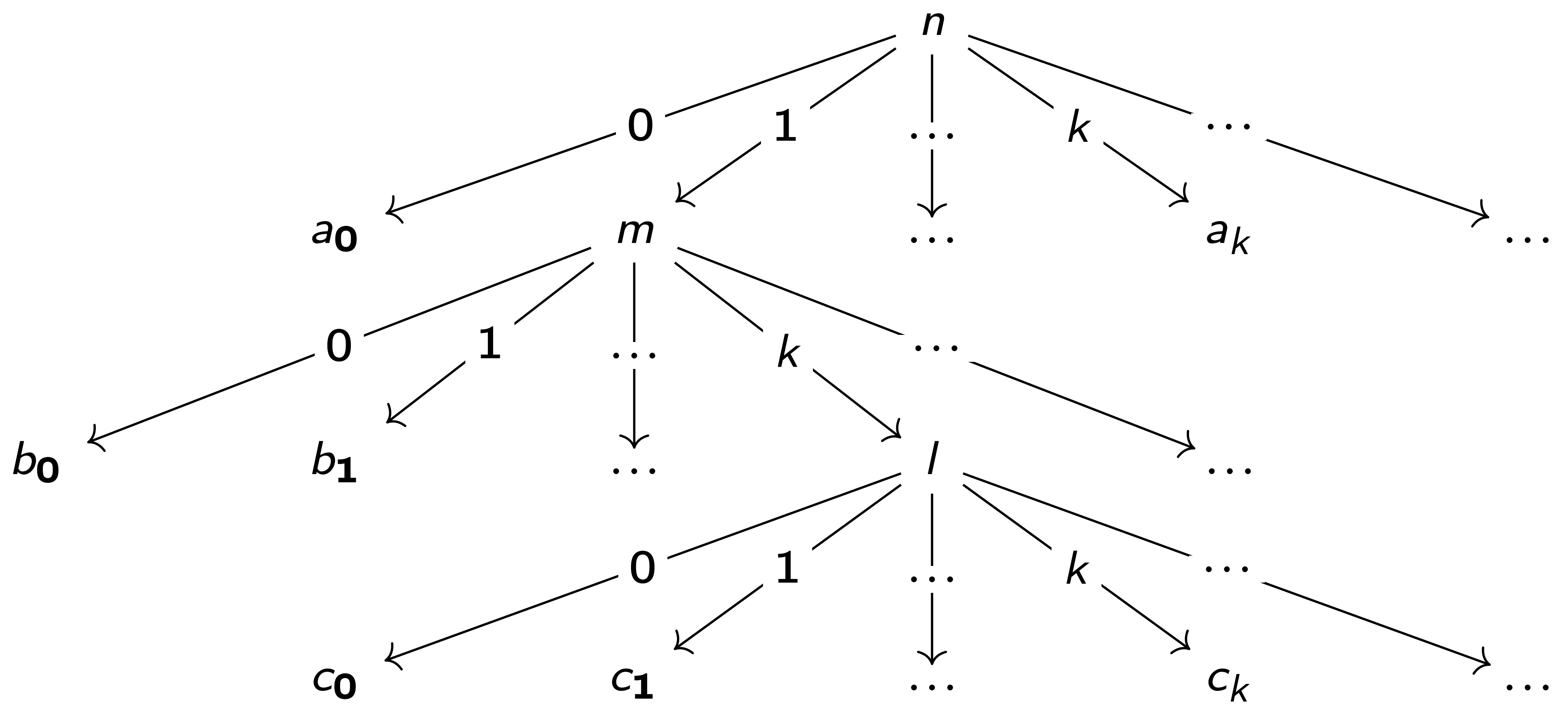


I. Continuity

For further information please ask the oracle

We consider the following ***decode*** *function :*

$$\begin{array}{lcl} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\, A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

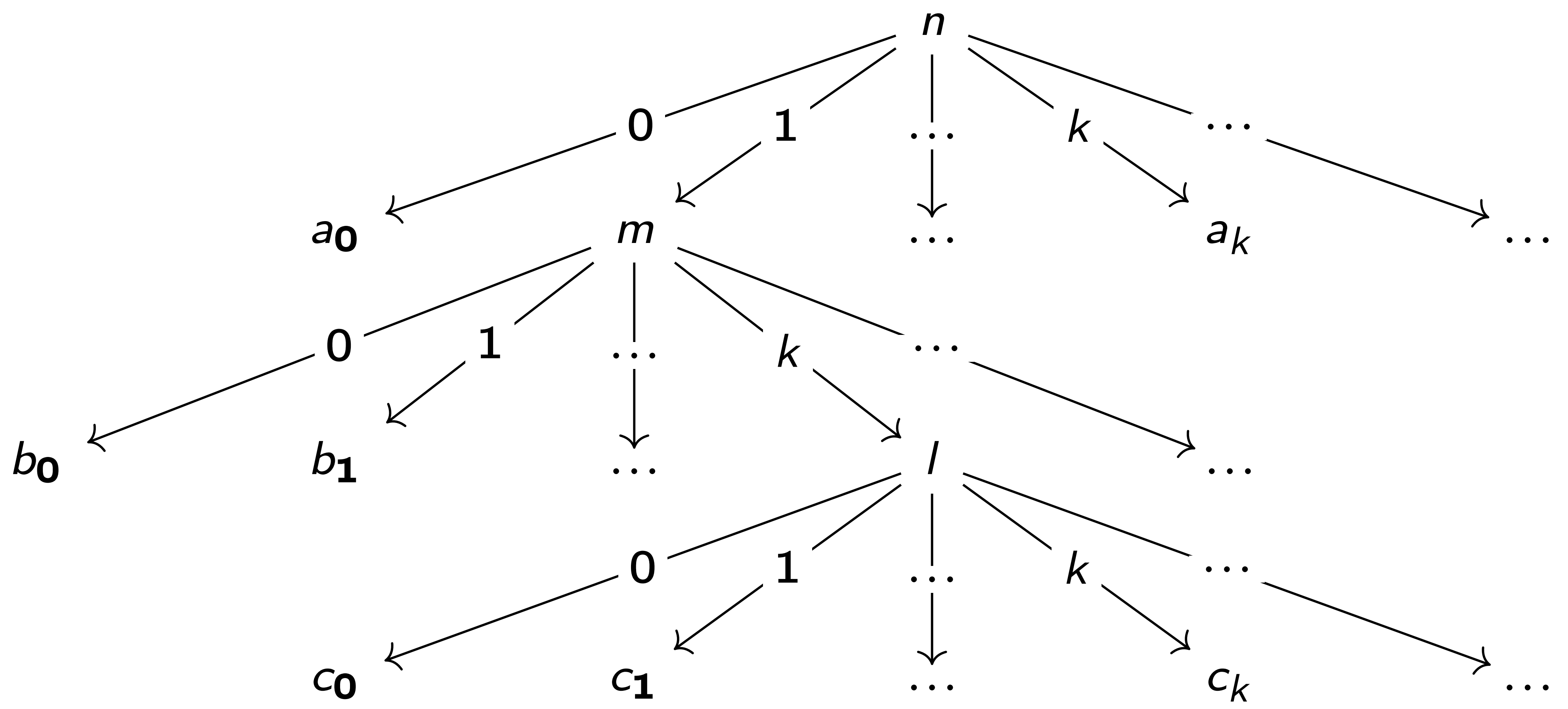


I. Continuity

For further information please ask the oracle

We consider the following ***decode*** *function :*

$$
\begin{array}{lcl}
\partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\, A \\
\partial\ \alpha\ (\eta\ x) & := & x \\
\partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i))
\end{array}
$$



I. Continuity

For further information please ask the oracle

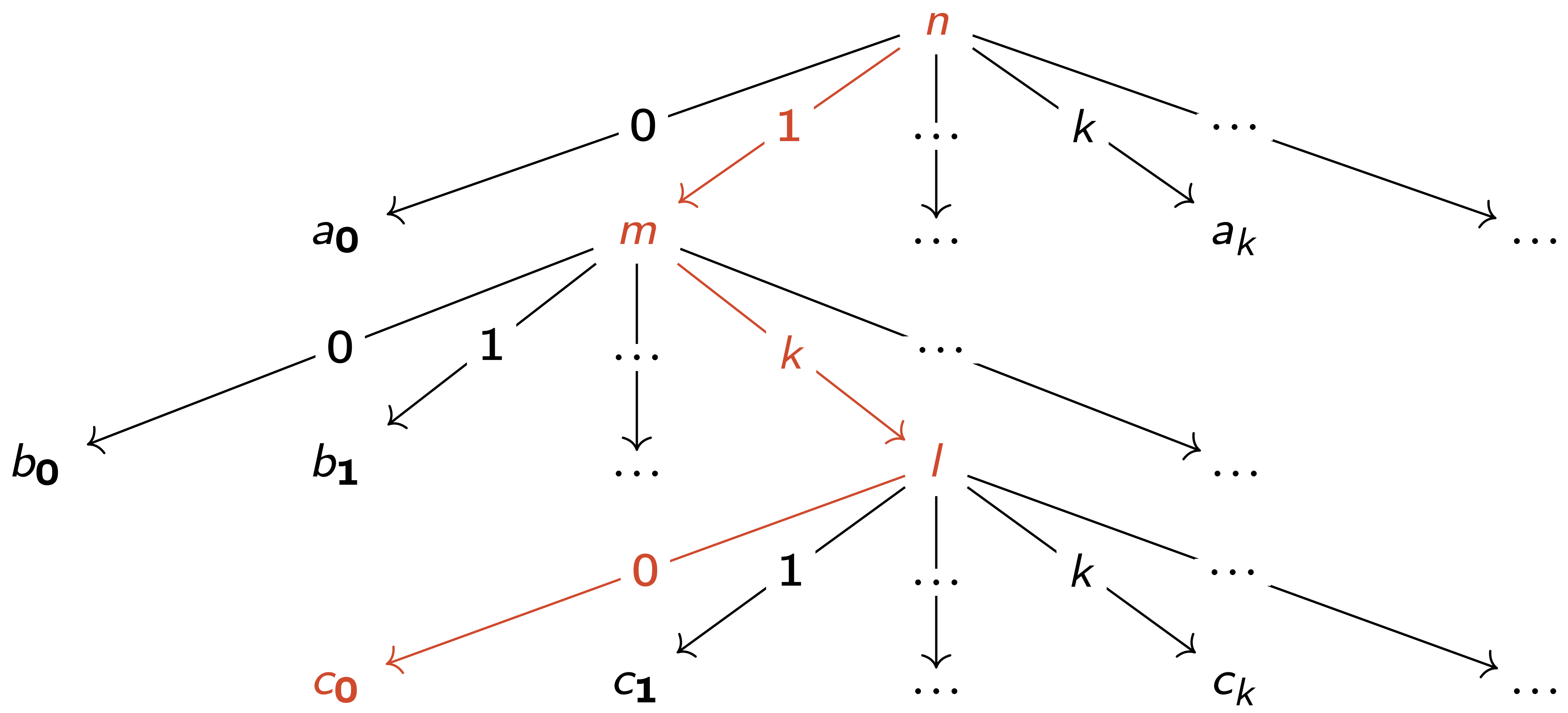
We consider the following ***decode*** *function :*

$\downarrow$ a

$\downarrow$ b

$$\begin{array}{lcl}
\partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\, A \\
\partial\ \alpha\ (\eta\ x) & := & x \\
\partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i))
\end{array}$$

$$\alpha\ n = 1$$



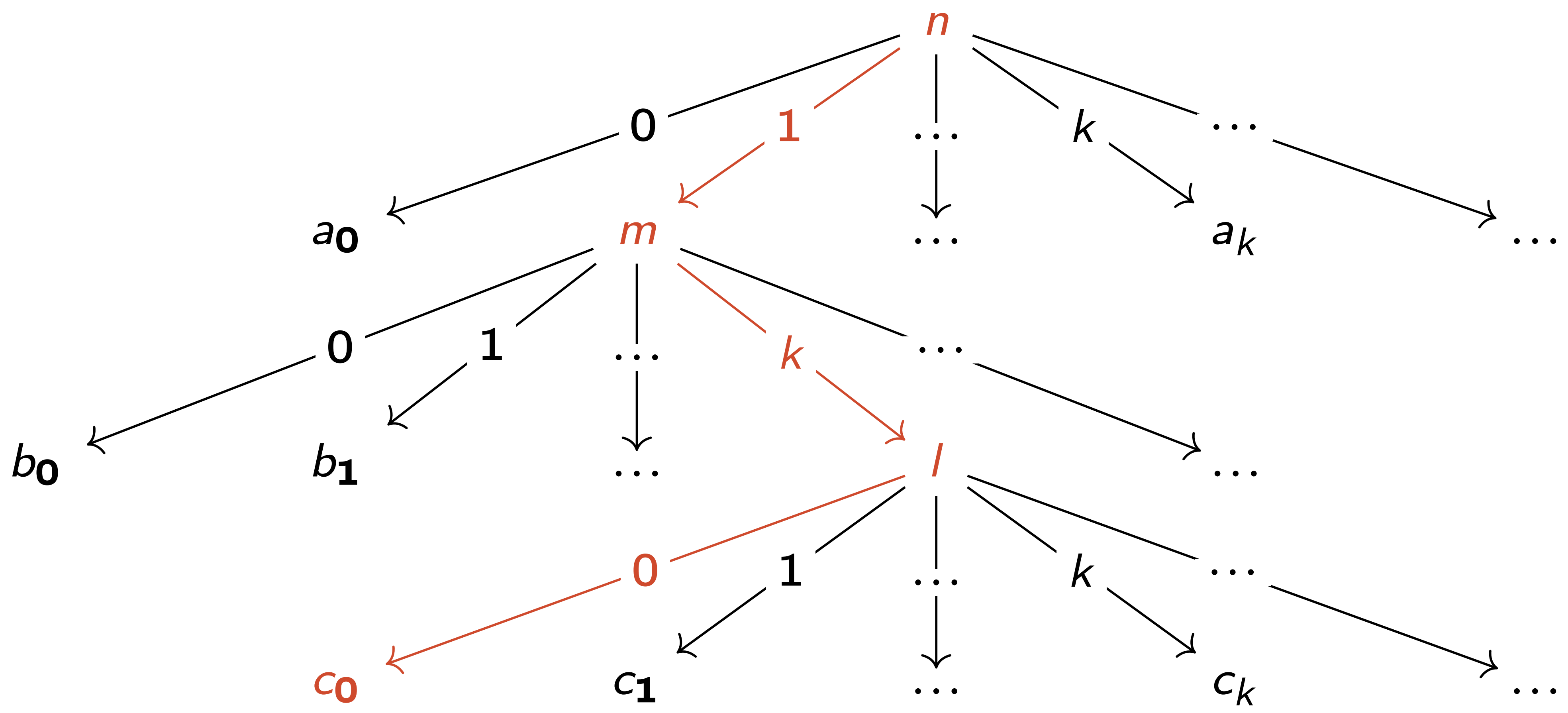
I. Continuity

For further information please ask the oracle

We consider the following ***decode*** *function :*

$$
\begin{array}{lll}
\partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\, A \\
\partial\ \alpha\ (\eta\ x) & := & x \\
\partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i))
\end{array}
$$

$$
\begin{array}{l}
\alpha\ n = 1 \\
\alpha\ m = k
\end{array}
$$



I. Continuity

For further information please ask the oracle

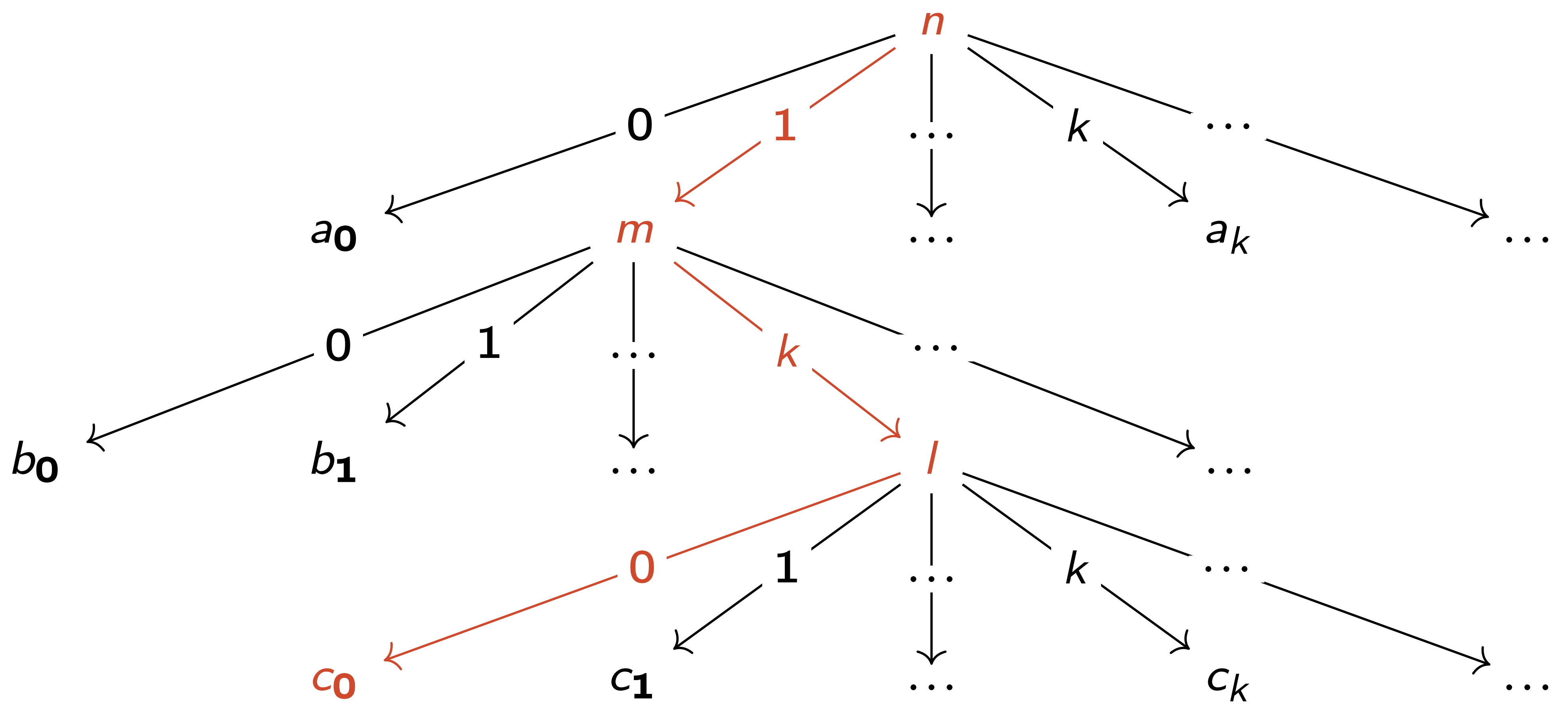
*We consider the following **decode** function :*

$\downarrow$ a

$\downarrow$ b

$$\begin{aligned} \partial \quad &: \quad \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\, A \\ \partial\ \alpha\ (\eta\ x) \quad &:= \quad x \\ \partial\ \alpha\ (\beta\ k\ i) \quad &:= \quad \partial\ \alpha\ (k\ (\alpha\ i)) \end{aligned}$$

$$\begin{aligned} \alpha\ n &= 1 \\ \alpha\ m &= k \\ \alpha\ l &= 0 \end{aligned}$$



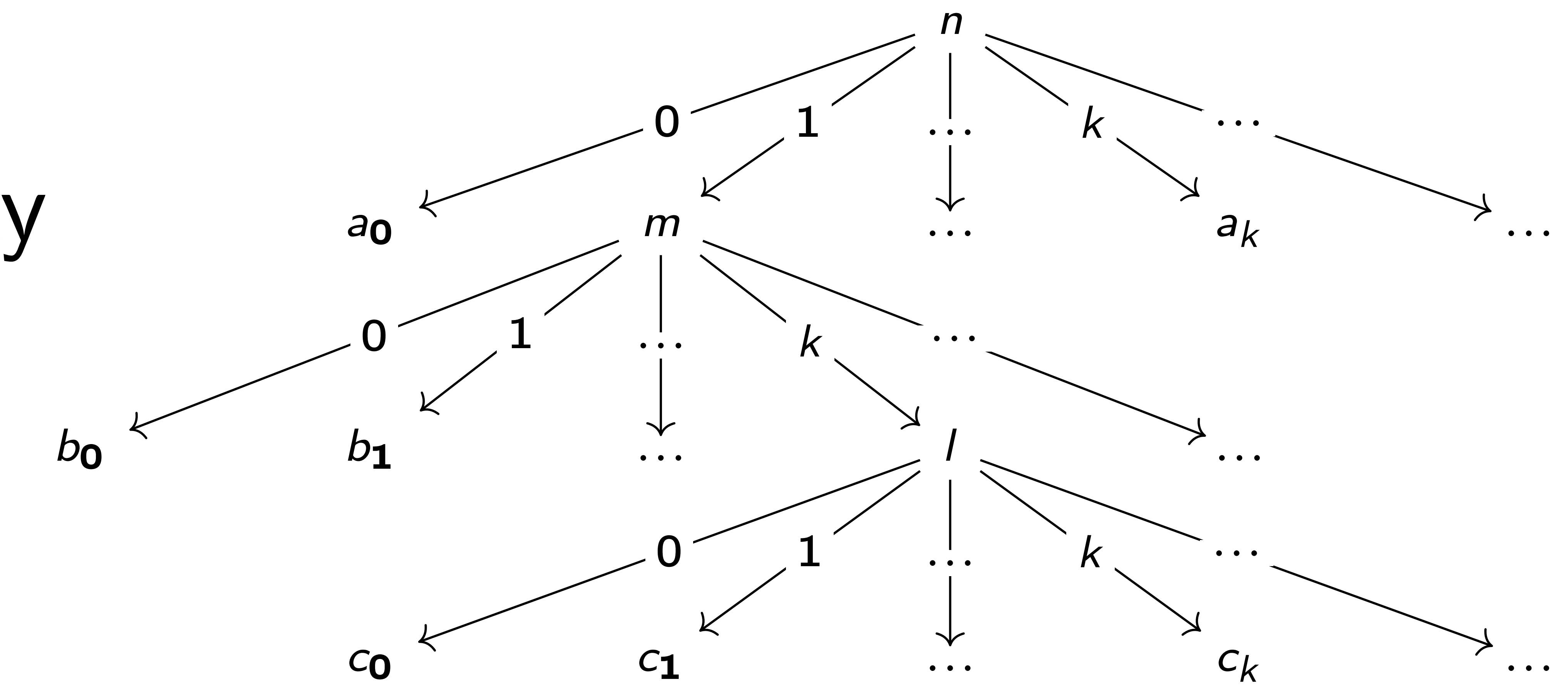
I. Continuity

Talking trees

*We consider the following **Dialogue** operator :*

$$
\begin{aligned}
&\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \quad := \\
&\quad \mid \quad \eta : A \to \mathfrak{D}\ A \\
&\quad \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.
\end{aligned}
$$

$(\mathfrak{D},\ \eta,\ \texttt{bind})$ is a monad up to extensionality



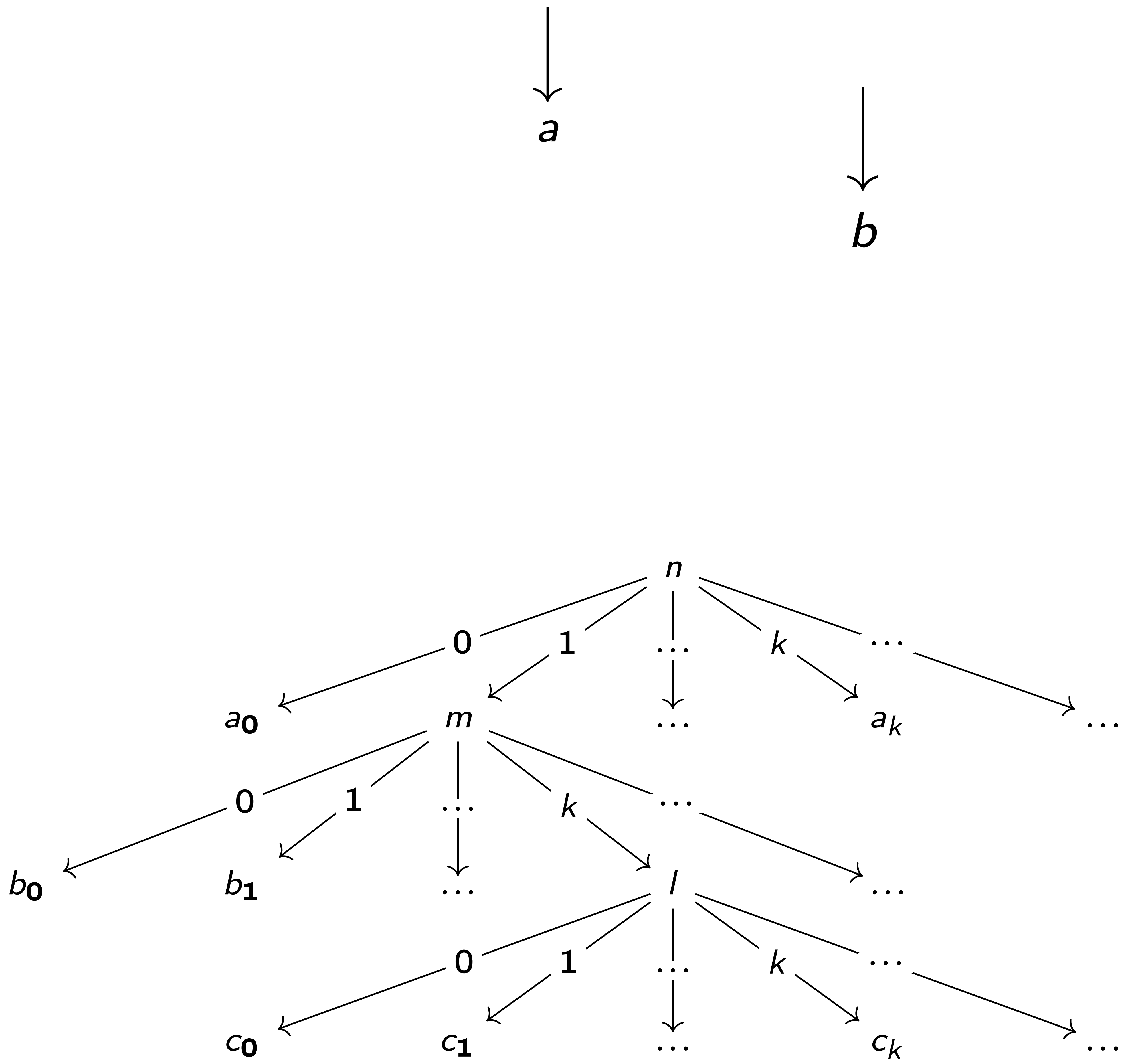
I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

$$\begin{array}{l}\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \qquad := \\ \quad | \quad \eta : A \to \mathfrak{D}\ A \\ \quad | \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.\end{array}$$

$(\mathfrak{D},\ \eta,\ \texttt{bind})$ is a *"moral"* monad



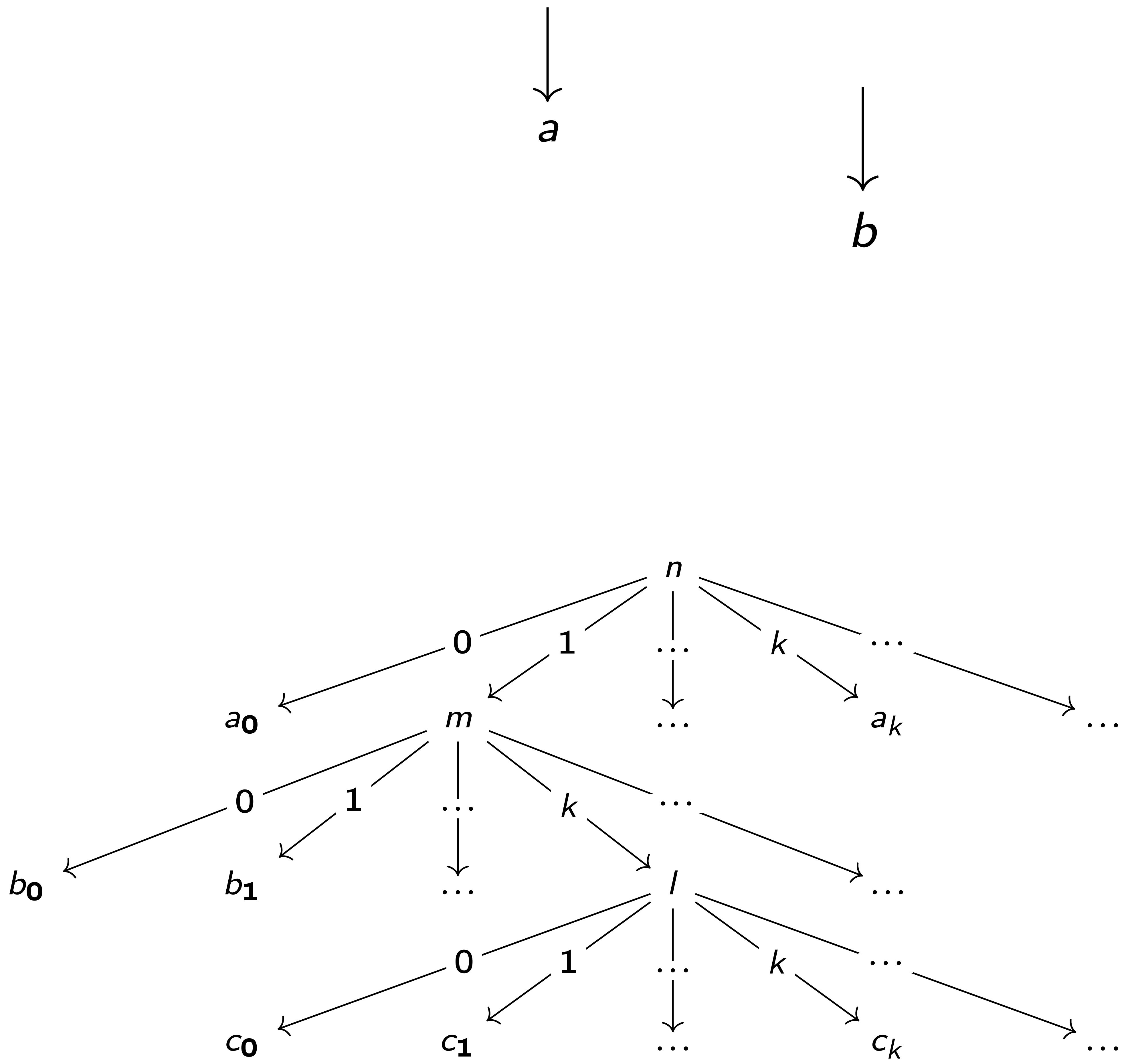
I. Continuity

Talking trees

We consider the following ***Dialogue*** *operator :*

$$\begin{array}{l} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \qquad := \\ \mid \quad \eta : A \to \mathfrak{D}\ A \\ \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. \end{array}$$

$(\mathfrak{D},\ \eta,\ \texttt{bind})$ is a ”*moral*” monad



→ *It is an effect!*

I. Continuity

For further information please ask the oracle

$$
\begin{array}{lll}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & & := \\
\mid & \eta : A \to \mathfrak{D}\ A & \\
\mid & \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. &
\end{array}
$$

Definition

A function $f : (\mathbb{N} \to \mathbb{N}) \to A$ is said **continuous** if :

$$\exists d : \mathfrak{D} A.\ \forall \alpha : \mathbb{N} \to \mathbb{N}.\ f\ \alpha = \partial\ d\ \alpha$$

I. Continuity

Continuity... Continuity everywhere

Folklore result

Any computable function is continuous

I. Continuity

Continuity... Continuity everywhere

Folklore result

Any computable function is continuous

Theorem

Any function $\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is continuous

I. Continuity

Continuity... Continuity everywhere

Folklore result

Any computable function is continuous

Theorem

Any function $\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is continuous

Smaller theorem

Any function $\vdash_{\mathsf{T}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is continuous

II. The case of System T

Type theory for beginners

II. The case of System T

System T

$$\frac{}{\Gamma, x : A \vdash_{\mathrm{T}} x : A} \qquad \frac{}{\Gamma \vdash_{\mathrm{T}} \mathrm{z} : \mathrm{N}} \qquad \frac{\Gamma \vdash_{\mathrm{T}} t : \mathrm{N}}{\Gamma \vdash_{\mathrm{T}} \mathrm{succ}\ t : \mathrm{N}}$$

$$\frac{\Gamma \vdash_{\mathrm{T}} t : A \to B \qquad \Gamma \vdash_{\mathrm{T}} u : A}{\Gamma \vdash_{\mathrm{T}} t\ u : B} \qquad \frac{\Gamma, x : A \vdash_{\mathrm{T}} t : B}{\Gamma \vdash_{\mathrm{T}} \lambda x.\ t : A \to B}$$

$$\frac{\Gamma \vdash_{\mathrm{T}} t : \mathrm{N} \qquad \Gamma \vdash_{\mathrm{T}} u : A \qquad \Gamma \vdash_{\mathrm{T}} v : A \to \mathrm{N} \to A}{\Gamma \vdash_{\mathrm{T}} \mathrm{rec}\ t\ u\ v : A}$$

II. The case of System T

System T + oracle

$$\frac{}{\Gamma,\ x : A \vdash_{\mathrm{T}}\ x : A} \qquad \frac{}{\Gamma \vdash_{\mathrm{T}}\ \mathrm{z} : \mathrm{N}} \qquad \frac{\Gamma \vdash_{\mathrm{T}} t : \mathrm{N}}{\Gamma \vdash_{\mathrm{T}}\ \mathrm{succ}\ t : \mathrm{N}}$$

$$\frac{\Gamma \vdash_{\mathrm{T}}\ t : A \to B \qquad \Gamma \vdash_{\mathrm{T}}\ u : A}{\Gamma \vdash_{\mathrm{T}}\ t\ u : B} \qquad \frac{\Gamma,\ x : A \vdash_{\mathrm{T}}\ t : B}{\Gamma \vdash_{\mathrm{T}}\ \lambda x.\ t : A \to B}$$

$$\frac{\Gamma \vdash_{\mathrm{T}}\ t : \mathrm{N} \qquad \Gamma \vdash_{\mathrm{T}}\ u : A \qquad \Gamma \vdash_{\mathrm{T}}\ v : A \to \mathrm{N} \to A}{\Gamma \vdash_{\mathrm{T}} \mathrm{rec}\ t\ u\ v : A} \qquad \frac{}{\Gamma \vdash_{\mathrm{T}}\ \alpha : \mathrm{N} \to \mathrm{N}}$$

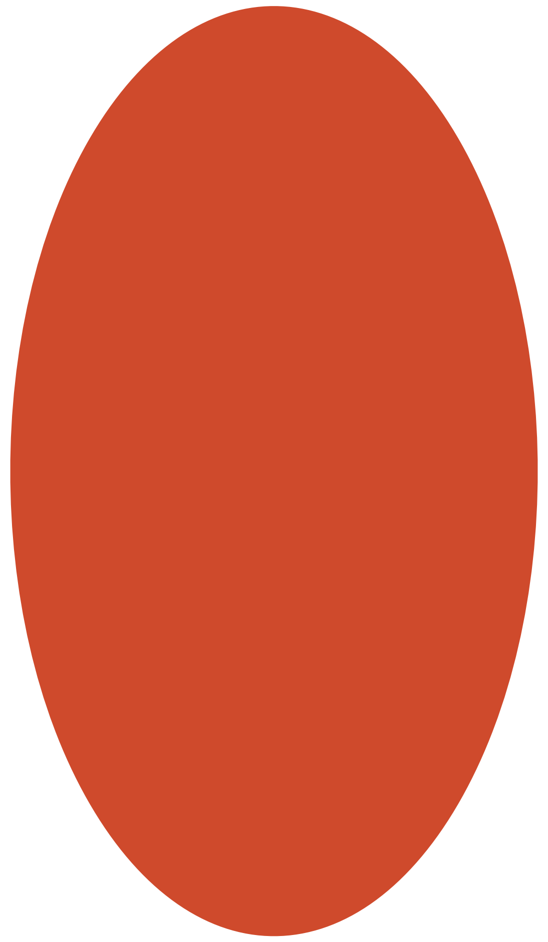
II. The case of System T

System T + oracle

$$\begin{tikzcd}
\alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} t : \mathsf{N} \arrow[d, "???"] \\
\vdash_{\texttt{metatheory}} d_t : \mathfrak{D}\,\mathbb{N}
\end{tikzcd}$$

II. The case of System T

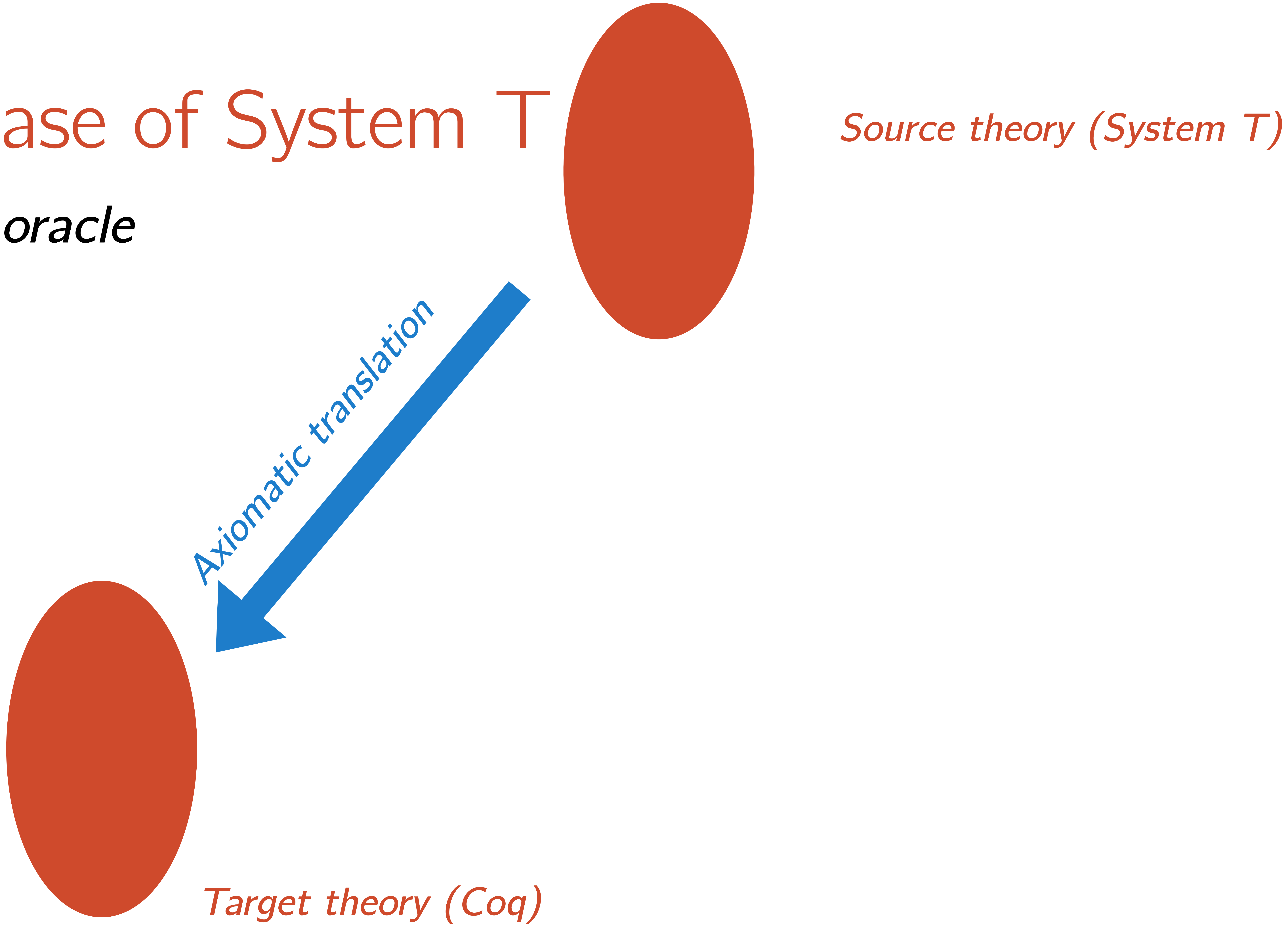
System T + oracle



Source theory (System T)

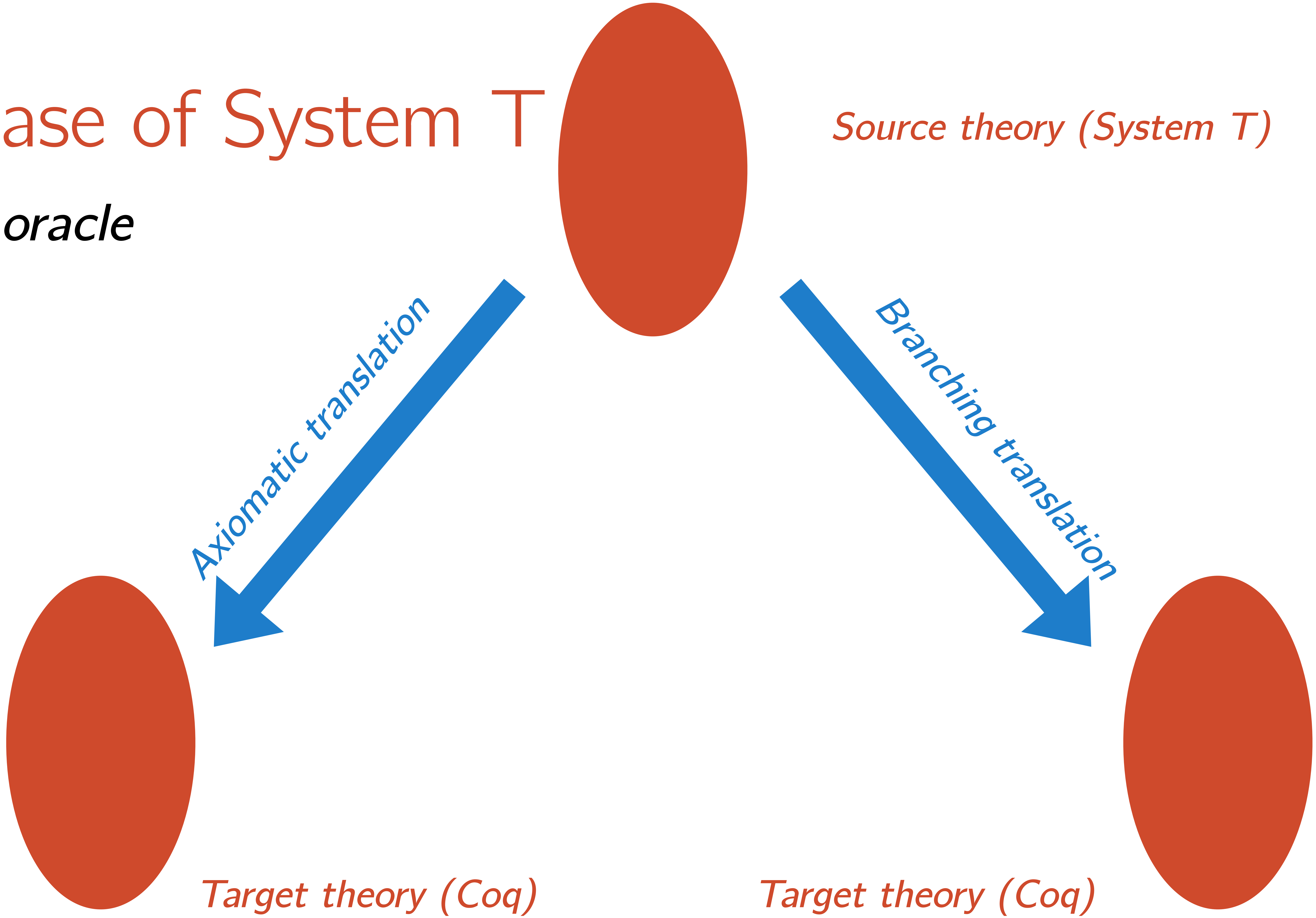
II. The case of System T

System T + oracle



II. The case of System T

System T + oracle



Source theory (System T)

Axiomatic translation

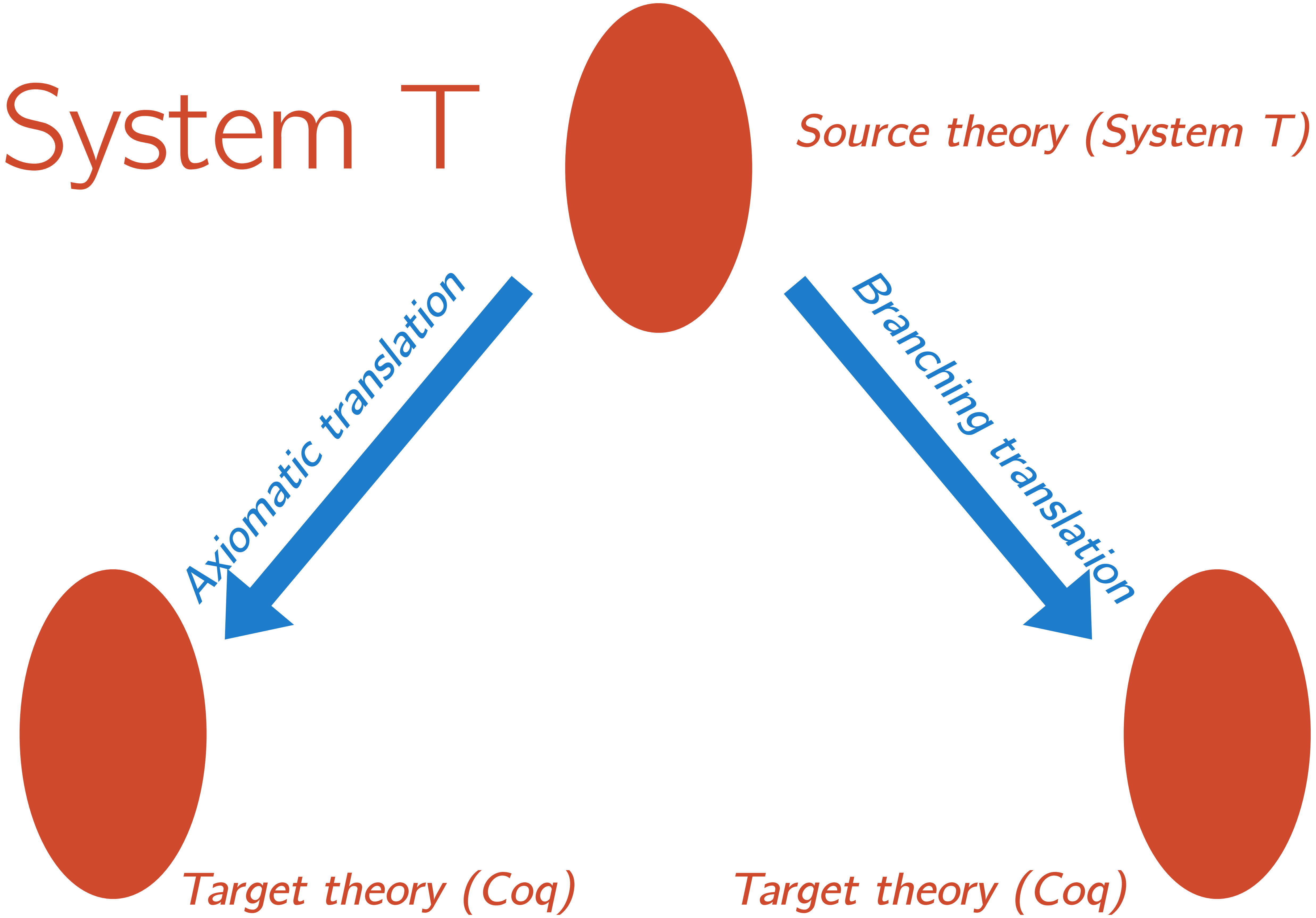
Branching translation

Target theory (Coq)

Target theory (Coq)

II. The case of System T

System T + oracle

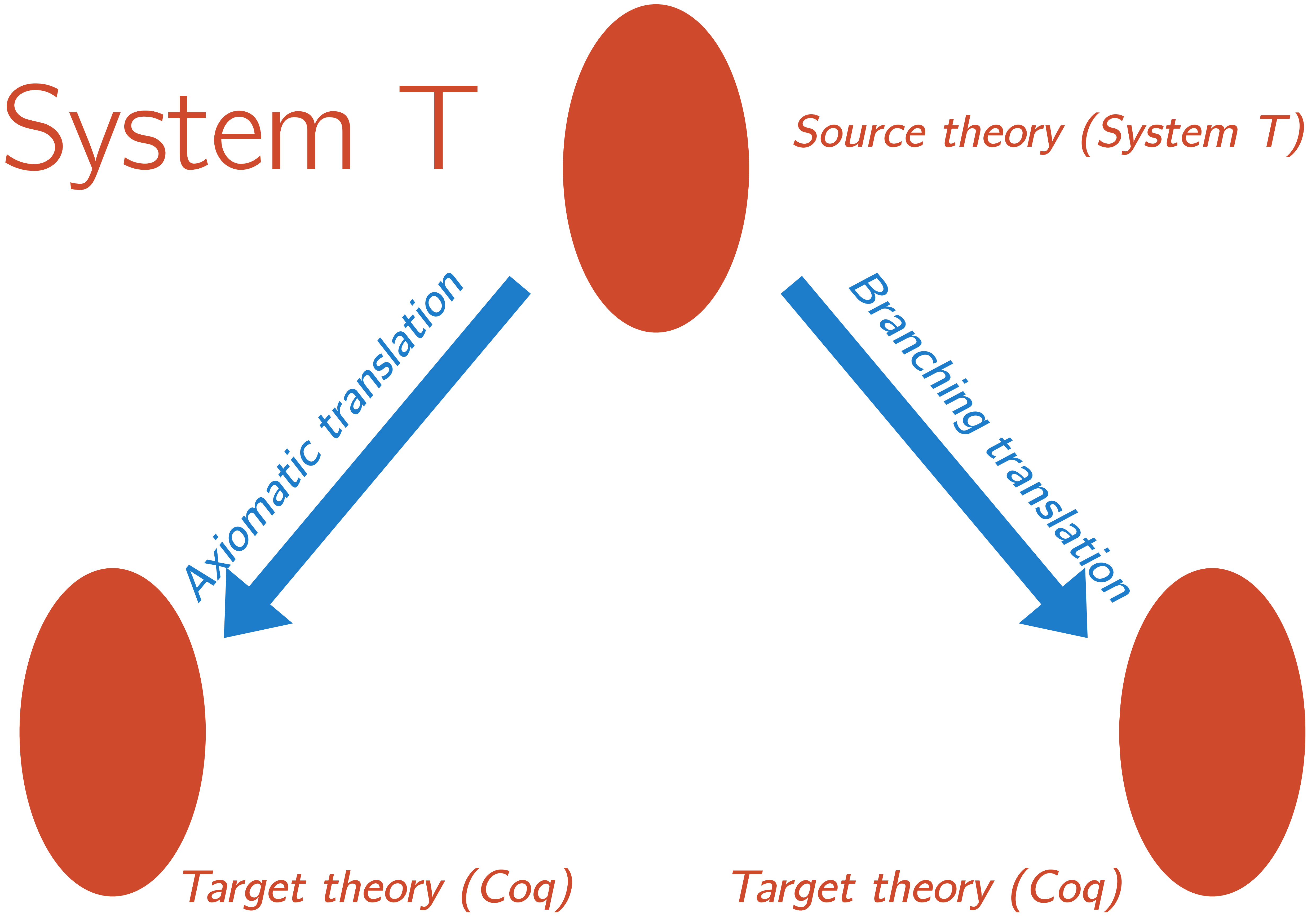


$\alpha : \mathbb{N} \to \mathbb{N} \vdash_{\mathsf{CIC}} t : \mathbb{N}$

$\vdash_{\mathsf{CIC}} d : \mathfrak{D}\ \mathbb{N}$

II. The case of System T

System T + oracle



$$\alpha : \mathbb{N} \to \mathbb{N} \vdash_{\mathsf{CIC}} t : \mathbb{N} \qquad \vdash_{\mathsf{CIC}} d : \mathfrak{D}\ \mathbb{N}$$

Kind of logical relation

II. The case of System T

Definition

For $\mathcal{S}$ and $\mathcal{T}$ two type theories, a **syntactic model** of $\mathcal{S}$ in $\mathcal{T}$ is:

- a translation $[\_]$ of terms of $\mathcal{S}$ into terms of $\mathcal{T}$;
- a translation $[\![\_]\!]$ of types of $\mathcal{S}$ into types of $\mathcal{T}$;
- a translation $[\![\_]\!]$ of contexts of $\mathcal{S}$ into contexts of $\mathcal{T}$;

II. The case of System T

Definition

For $\mathcal{S}$ and $\mathcal{T}$ two type theories, a **syntactic model** of $\mathcal{S}$ in $\mathcal{T}$ is:

- a translation $[\_]$ of terms of $\mathcal{S}$ into terms of $\mathcal{T}$;
- a translation $[\![\_]\!]$ of types of $\mathcal{S}$ into types of $\mathcal{T}$;
- a translation $[\![\_]\!]$ of contexts of $\mathcal{S}$ into contexts of $\mathcal{T}$;

In our setting, $\mathcal{S} := \mathrm{T}$, $\mathcal{T} := \mathrm{CIC}$ and the translations will be defined by induction on the syntax of T

II. The case of System T

The Axiomatic Translation

Given $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ *in* CIC, *we define the* ***Axiomatic Translation****:*

$$\Gamma \vdash_{\mathsf{T}} t : A$$

II. The case of System T

The Axiomatic Translation

Given $\alpha : \mathbb{N} \to \mathbb{N}$ in CIC, *we define the **Axiomatic Translation**:*

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Axiomatic}}$$

II. The case of System T

The Axiomatic Translation

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the* ***Axiomatic Translation****:*

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Axiomatic}} [\![\Gamma]\!]_a^\alpha \vdash_{\mathsf{CIC}} [t]_a^\alpha : [\![A]\!]_a^\alpha$$

II. The case of System T

The Axiomatic Translation

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\text{Translation}]{\text{Axiomatic}} \llbracket \Gamma \rrbracket_a^\alpha \vdash_{\mathsf{CIC}} [t]_a^\alpha : \llbracket A \rrbracket_a^\alpha$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the Axiomatic Translation:*

$$\begin{array}{lcl}
\mathsf{N} & \longrightarrow & \llbracket \mathsf{N} \rrbracket_a^\alpha := \mathbb{N} \\
A \to B & \longrightarrow & \llbracket A \rrbracket_a^\alpha \to \llbracket B \rrbracket_a^\alpha \\
x : A & \longrightarrow & x : \llbracket A \rrbracket_a^\alpha \\
\lambda x.\ t : A \to B & \longrightarrow & \lambda x.\ [t]_a^\alpha : \llbracket A \rrbracket_a^\alpha \to \llbracket B \rrbracket_a^\alpha \\
t\ \ u & \longrightarrow & [t]_a^\alpha\ [u]_a^\alpha \\
\mathsf{z} : \mathsf{N} & \longrightarrow & 0 : \mathbb{N} \\
\mathsf{succ} : \mathsf{N} \to \mathsf{N} & \longrightarrow & \mathsf{S} : \mathbb{N} \to \mathbb{N} \\
\mathsf{rec} : \mathsf{N} \to A \to (A \to \mathsf{N} \to A) \to A & \longrightarrow & \mathbb{N}\_\mathsf{rect}\ (\lambda \_.\ \llbracket A \rrbracket_a^\alpha)
\end{array}$$

II. The case of System T

The Axiomatic Translation

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Axiomatic}} [\![\Gamma]\!]_a^\alpha \vdash_{\mathsf{CIC}} [t]_a^\alpha : [\![A]\!]_a^\alpha$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the Axiomatic Translation:*

$$\begin{array}{lcl} . & \longrightarrow & \alpha : \mathbb{N} \to \mathbb{N} \\ \Gamma, x : A & \longrightarrow & [\![\Gamma]\!]_a^\alpha,\ x : [\![A]\!]_a^\alpha \\ \alpha : \mathsf{N} \to \mathsf{N} & \longrightarrow & \alpha \end{array}$$

II. The case of System T

The Axiomatic Translation

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Axiomatic}} [\![\Gamma]\!]_a^\alpha \vdash_{\mathsf{CIC}} [t]_a^\alpha : [\![A]\!]_a^\alpha$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ in **CIC**, *we define the Axiomatic Translation:*

$$\begin{array}{lcl} . & \longrightarrow & \alpha : \mathbb{N} \to \mathbb{N} \\ \Gamma, x : A & \longrightarrow & [\![\Gamma]\!]_a^\alpha,\ x : [\![A]\!]_a^\alpha \\ \alpha : \mathsf{N} \to \mathsf{N} & \longrightarrow & \alpha \end{array}$$

Theorem

We have the following properties:

- **Computational soundness:** $M \equiv N$ implies $[M]_a^\alpha \equiv [N]_a^\alpha$
- **Typing soundness:** $\Gamma \vdash_{\mathsf{T}} M : A$ implies $[\![\Gamma]\!]_a^\alpha \vdash_{\mathsf{CIC}} [M]_a^\alpha : [\![A]\!]_a^\alpha$

II. The case of System T

System Tree

We define the ***Branching Translation****:*

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\text{Translation}]{\text{Branching}} [\![\Gamma]\!]_b \vdash_{\mathsf{CIC}} [t]_b : [\![A]\!]_b$$

II. The case of System T

$$\Gamma \vdash_{\mathrm{T}} t : A \xrightarrow[\text{Translation}]{\text{Branching}} [\![\Gamma]\!]_b \vdash_{\mathrm{CIC}} [t]_b : [\![A]\!]_b$$

System Tree

*We define the **Branching Translation**:*

$$\begin{array}{lcl}
\mathrm{N} & \longrightarrow & \mathfrak{D}\ \mathbb{N} \\
A \to B & \longrightarrow & [\![A]\!]_b \to [\![B]\!]_b \\
x : A & \longrightarrow & x_b : [\![A]\!]_b \\
\lambda x.\ t : A \to B & \longrightarrow & \lambda x.\ [t]_b : [\![A]\!]_b \to [\![B]\!] \\
t\ \ u & \longrightarrow & [t]_b\ [u]_b \\
\mathrm{z} : \mathrm{N} & \longrightarrow & \eta\ 0 : \mathfrak{D}\ \mathbb{N} \\
\mathrm{succ} : \mathrm{N} \to \mathrm{N} & \longrightarrow & \mathtt{map}\ \mathrm{S} : \mathfrak{D}\ \mathbb{N} \to \mathfrak{D}\ \mathbb{N} \\
\mathrm{rec} : \mathrm{N} \to A \to (A \to \mathrm{N} \to A) \to A & \longrightarrow &
\end{array}$$

$$\lambda(u : [\![A]\!]_b)(v : [\![A]\!]_b \to [\![\mathbb{N}]\!]_b \to [\![A]\!]_b).$$

$$\mathtt{bind}\ (\mathbb{N}\_\mathrm{rect}\ (\lambda\_.\ [\![A]\!]_b)\ u\ (\lambda n\ a.\ v\ a\ (\eta\ n)))$$

II. The case of System T

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Branching}} [\![\Gamma]\!]_b \vdash_{\mathsf{CIC}} [t]_b : [\![A]\!]_b$$

System Tree

We define the ***Branching Translation****:*

$$
\begin{array}{lcl}
. & \longrightarrow & . \\
\Gamma, x : A & \longrightarrow & [\![\Gamma]\!]_b,\ x : [\![A]\!]_b \\
\alpha : \mathsf{N} \to \mathsf{N} & \longrightarrow & ???
\end{array}
$$

II. The case of System T

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Branching}} [\![\Gamma]\!]_b \vdash_{\mathsf{CIC}} [t]_b : [\![A]\!]_b$$

System Tree

We define the ***Branching Translation****:*

$$\cdot \longrightarrow \cdot$$

$$\Gamma, x : A \longrightarrow [\![\Gamma]\!]_b,\ x : [\![A]\!]_b$$

$$\alpha : \mathsf{N} \to \mathsf{N} \longrightarrow \gamma$$

II. The case of System T

$$\Gamma \vdash_T t : A \xrightarrow[\textit{Translation}]{\textit{Branching}} [\![\Gamma]\!]_b \vdash_{CIC} [t]_b : [\![A]\!]_b$$

System Tree

*We define the **Branching Translation**:*

$$\begin{array}{lcl} \cdot & \longrightarrow & \cdot \\ \Gamma, x : A & \longrightarrow & [\![\Gamma]\!]_b,\ x : [\![A]\!]_b \\ \alpha : \mathrm{N} \to \mathrm{N} & \longrightarrow & \gamma \end{array}$$

Theorem

We have the following properties:

- **Computational soundness:** $M \equiv N$ implies $[M]_b \equiv [N]_b$
- **Typing soundness:** $\Gamma \vdash_T M : A$ implies $[\![\Gamma]\!]_b \vdash_{CIC} [M]_b : [\![A]\!]_b$

II. The case of System T

$$\alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} t : \mathsf{N}$$

II. The case of System T

$$\begin{tikzcd}
& \alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} t : \mathsf{N} \arrow[dl] \arrow[dr] & \\
\alpha : \mathbb{N} \to \mathbb{N} \vdash_{\mathsf{CIC}} [t]^{\alpha}_{a} : \mathbb{N} & & \vdash_{\mathsf{CIC}} [t]_{b} : \mathfrak{D}\ \mathbb{N}
\end{tikzcd}$$

II. The case of System T

$$\begin{tikzcd}
 & \alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} t : \mathsf{N} \arrow[dl] \arrow[dr] & \\
\alpha : \mathbb{N} \to \mathbb{N} \vdash_{\mathsf{CIC}} [t]^{\alpha}_{a} : \mathbb{N} & & \vdash_{\mathsf{CIC}} [t]_{b} : \mathfrak{D}\ \mathbb{N}
\end{tikzcd}$$

We want to guarantee: $\forall \alpha : \mathbb{N} \to \mathbb{N}.\ [t]^{\alpha}_{a} = \partial\ [t]_{b}\ \alpha$

II. The case of System T

A translation to bind them all

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the* ***Parametricity Translation****:*

$$A\ \texttt{type} \xrightarrow[\textit{Translation}]{\textit{Parametricity}} [\![A]\!]^{\alpha}_{\epsilon} : [\![A]\!]^{\alpha}_{a} \to [\![A]\!]_{b} \to \square$$

II. The case of System T

A translation to bind them all

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the* ***Parametricity Translation****:*

$$A\ \mathtt{type} \quad \xrightarrow[\text{Translation}]{\text{Parametricity}} \quad \llbracket A \rrbracket^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{a} \to \llbracket A \rrbracket_{b} \to \square$$

$$\Gamma \vdash_{\mathrm{T}} t : A \quad \xrightarrow[\text{Translation}]{\text{Parametricity}} \quad \llbracket \Gamma \rrbracket^{\alpha}_{\epsilon} \vdash_{\mathrm{CIC}} [t]^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{\epsilon}\ [t]^{\alpha}_{a}\ [t]_{b}$$

II. The case of System T

A translation to bind them all

$$A\ \texttt{type} \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket A \rrbracket^\alpha_\epsilon : \llbracket A \rrbracket^\alpha_a \to \llbracket A \rrbracket_b \to \square$$

$$\Gamma \vdash_{\mathrm{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket \Gamma \rrbracket^\alpha_\epsilon \vdash_{\mathrm{CIC}} [t]^\alpha_\epsilon : \llbracket A \rrbracket^\alpha_\epsilon\ [t]^\alpha_a\ [t]_b$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* CIC, *we define the* ***Parametricity Translation****:*

$$\begin{aligned}
\llbracket \mathrm{N} \rrbracket^\alpha_\epsilon\ n\ n_b &:= n = \partial\ n_b\ \alpha \\
\llbracket A \to B \rrbracket_\epsilon\ f\ f_b &:= \forall x\ x_b.\ (\llbracket A \rrbracket^\alpha_\epsilon\ x\ x_b) \to \llbracket B \rrbracket^\alpha_\epsilon\ (f\ x)\ (f_b\ x_b) \\
[x : A]^\alpha_\epsilon &:= x_\epsilon : \llbracket A \rrbracket^\alpha_\epsilon\ x\ x_b \\
[\lambda x.\ t : A \to B]^\alpha_\epsilon &:= \lambda(x : \llbracket A \rrbracket^\alpha_a)(x_b : \llbracket A \rrbracket_b)(x_\epsilon : [A]^\alpha_\epsilon\ x\ x_b).\ [t]^\alpha_\epsilon \\
[t\ u : B]^\alpha_\epsilon &:= [t]^\alpha_\epsilon\ [u]^\alpha_a\ [u]_b\ [u]^\alpha_\epsilon \\
[z : \mathrm{N}]^\alpha_\epsilon &:= \mathrm{refl}\ 0 \\
[\mathrm{succ} : \mathrm{N} \to \mathrm{N}]^\alpha_\epsilon &:= \texttt{succ\_lemma} \\
[\mathrm{rec}]^\alpha_\epsilon : &:= \texttt{rec\_lemma}
\end{aligned}$$

II. The case of System T

A translation to bind them all

$$A \ \texttt{type} \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket A \rrbracket^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{a} \to \llbracket A \rrbracket_{b} \to \square$$

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket \Gamma \rrbracket^{\alpha}_{\epsilon} \vdash_{\mathsf{CIC}} [t]^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{\epsilon} \ [t]^{\alpha}_{a} \ [t]_{b}$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* **CIC**, *we define the* ***Parametricity Translation***:

$$\llbracket . \rrbracket^{\alpha}_{\epsilon} \quad := \quad \alpha : \mathbb{N} \to \mathbb{N}$$

$$\llbracket \Gamma, x : A \rrbracket^{\alpha}_{\epsilon} \quad := \quad \llbracket \Gamma \rrbracket^{\alpha}_{\epsilon}, \ x : \llbracket A \rrbracket^{\alpha}_{a}, \ x_b : \llbracket A \rrbracket_{b}, \ x_{\epsilon} : [A]^{\alpha}_{\epsilon} \ x \ x_b$$

$$[\alpha : \mathsf{N} \to \mathsf{N}]^{\alpha}_{\epsilon} \quad := \quad \gamma_{\epsilon}$$

II. The case of System T

A translation to bind them all

$$A\ \texttt{type} \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket A \rrbracket^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{a} \to \llbracket A \rrbracket_{b} \to \square$$

$$\Gamma \vdash_{\mathsf{T}} t : A \xrightarrow[\textit{Translation}]{\textit{Parametricity}} \llbracket \Gamma \rrbracket^{\alpha}_{\epsilon} \vdash_{\mathsf{CIC}} [t]^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{\epsilon}\ [t]^{\alpha}_{a}\ [t]_{b}$$

Given $\alpha : \mathbb{N} \to \mathbb{N}$ *in* **CIC**, *we define the* ***Parametricity Translation****:*

$$\begin{array}{lcl}
\llbracket . \rrbracket^{\alpha}_{\epsilon} & := & \alpha : \mathbb{N} \to \mathbb{N} \\
\llbracket \Gamma, x : A \rrbracket^{\alpha}_{\epsilon} & := & \llbracket \Gamma \rrbracket^{\alpha}_{\epsilon},\ x : \llbracket A \rrbracket^{\alpha}_{a},\ x_b : \llbracket A \rrbracket_{b},\ x_{\epsilon} : [A]^{\alpha}_{\epsilon}\ x\ x_b \\
[\alpha : \mathrm{N} \to \mathrm{N}]^{\alpha}_{\epsilon} & := & \gamma_{\epsilon}
\end{array}$$

Theorem

We have the following properties:

- **Computational soundness:** $M \equiv N$ implies $[M]^{\alpha}_{\epsilon} \equiv [N]^{\alpha}_{\epsilon}$
- **Typing soundness:** $\Gamma \vdash_{\mathsf{T}} M : A$ implies $\llbracket \Gamma \rrbracket^{\alpha}_{\epsilon} \vdash_{\mathsf{CIC}} [M]^{\alpha}_{\epsilon} : \llbracket A \rrbracket^{\alpha}_{\epsilon}\ [M]^{\alpha}_{a}\ [M]_{b}$

II. The case of System T

First Theorem

We have the following:

Theorem

Any function $\vdash_{\mathsf{T}} f : (\mathsf{N} \to \mathsf{N}) \to \mathsf{N}$ is continuous

II. The case of System T

First Theorem

We have the following:

$$\vdash_{\mathsf{T}} f : (\mathsf{N} \to \mathsf{N}) \to \mathsf{N}$$

II. The case of System T

First Theorem

We have the following:

$$\vdash_{\mathsf{T}} f : (\mathsf{N} \to \mathsf{N}) \to \mathsf{N}$$

$$\Longrightarrow \quad \alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} f\ \alpha : \mathsf{N}$$

II. The case of System T

First Theorem

We have the following:

$$\vdash_{\mathsf{T}} f : (\mathsf{N} \to \mathsf{N}) \to \mathsf{N}$$

$$\Longrightarrow \quad \alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} f\ \alpha : \mathsf{N}$$

For any $\vdash_{\mathsf{CIC}} \alpha : \mathbb{N} \to \mathbb{N}$:

$$[f\ \alpha]^{\alpha}_{\epsilon} : [\![\mathsf{N}]\!]^{\alpha}_{\epsilon}\ [f\ \alpha]^{\alpha}_{a}\ [f\ \alpha]_{b}$$

II. The case of System T

First Theorem

We have the following:

$$\vdash_{\mathsf{T}} f : (\mathsf{N} \to \mathsf{N}) \to \mathsf{N}$$

$$\Longrightarrow \quad \alpha : \mathsf{N} \to \mathsf{N} \vdash_{\mathsf{T}} f\ \alpha : \mathsf{N}$$

For any $\vdash_{\mathsf{CIC}} \alpha : \mathbb{N} \to \mathbb{N}$:

$$[f\ \alpha]^{\alpha}_{\epsilon} : [\![\mathsf{N}]\!]^{\alpha}_{\epsilon}\ [f\ \alpha]^{\alpha}_{a}\ [f\ \alpha]_{b}$$

$$\Longrightarrow \quad [f\ \alpha]^{\alpha}_{\epsilon} : [f]^{\alpha}_{a}\ \alpha = \partial\ ([f]_{b}\ \gamma)\ \alpha$$

II. The case of System T

What about α ?

In the axiom translation

In the branching translation

$$\alpha : (\mathbb{N} \to \mathbb{N}) \vdash \alpha : (\mathbb{N} \to \mathbb{N}) \quad \Longleftrightarrow \quad \vdash^{?}_{\mathsf{CIC}} \gamma : (\mathfrak{D}\ \mathbb{N} \to \mathfrak{D}\ \mathbb{N})$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash^{?}_{\mathsf{CIC}} \gamma_{\varepsilon} : [\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \ \gamma$$

II. The case of System T

What about α ?

In the axiom translation

$$\alpha : (\mathbb{N} \to \mathbb{N}) \vdash \alpha : (\mathbb{N} \to \mathbb{N})$$

In the branching translation

$$\vdash^{?}_{\mathsf{CIC}} \gamma : (\mathfrak{D}\ \mathbb{N} \to \mathfrak{D}\ \mathbb{N})$$

$$\longleftrightarrow$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash^{?}_{\mathsf{CIC}} \gamma_{\varepsilon} : [\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \ \gamma$$

We want:

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma$$

II. The case of System T

What about α ?

In the axiom translation

$$\alpha : (\mathbb{N} \to \mathbb{N}) \vdash \alpha : (\mathbb{N} \to \mathbb{N})$$

In the branching translation

$$\vdash^{?}_{\mathsf{CIC}} \gamma : (\mathfrak{D}\ \mathbb{N} \to \mathfrak{D}\ \mathbb{N})$$

⟷

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash^{?}_{\mathsf{CIC}} \gamma_{\varepsilon} : [\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \ \gamma$$

We want:

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

II. The case of System T

What about α ? $[\![\mathsf{N} \to \mathsf{N}]\!]_{\epsilon}^{\alpha}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$

$$\begin{array}{ll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\ \mid \quad \eta : A \to \mathfrak{D}\ A & \\ \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & \end{array}$$

$$\begin{array}{lll} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\ A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]_{\epsilon}^{\alpha}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{ll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\ \quad \mid\quad \eta : A \to \mathfrak{D}\ A & \\ \quad \mid\quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & \end{array}$$

$$\begin{array}{lcl} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

For any numeral n:

$$\forall \alpha.\ n = \partial\ (\eta\ n)\ \alpha$$

II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{ll} \multicolumn{2}{l}{\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square \quad :=} \\ \mid & \eta : A \to \mathfrak{D}\ A \\ \mid & \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. \end{array}$$

$$\begin{array}{lll} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

For any numeral n:

$$\forall \alpha.\ n = \partial\ (\eta\ n)\ \alpha$$

$$\begin{tikzcd} & {} \arrow[d] \\ n \arrow[r, leftrightarrow] & n \end{tikzcd}$$

II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{lll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := & \\ \quad \mid \quad \eta : A \to \mathfrak{D}\ A & & \\ \quad \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & & \end{array}$$

$$\begin{array}{lcl} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

For any numeral n:

$$\begin{tikzcd} & \ \arrow[d] \\ n \arrow[r, leftrightarrow] & n \end{tikzcd}$$

$$\forall \alpha.\ n = \partial\ (\eta\ n)\ \alpha$$

$$\forall \alpha.\ \alpha\ n = \partial\ (\gamma\ (\eta\ n))\ \alpha$$

II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

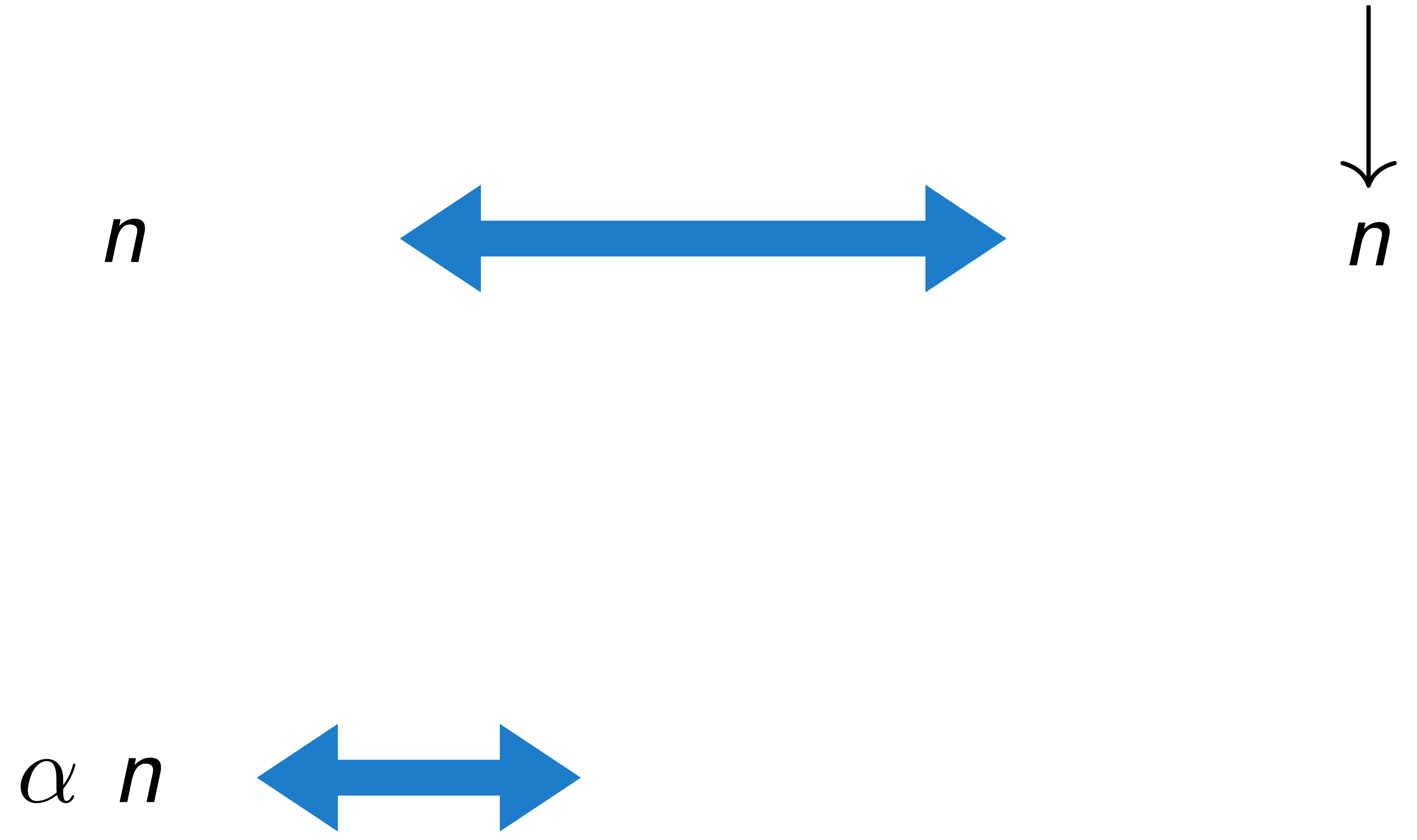
$$\begin{array}{ll}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\
\quad \mid\ \ \eta : A \to \mathfrak{D}\ A & \\
\quad \mid\ \ \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. &
\end{array}$$

$$\begin{array}{lcl}
\partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\
\partial\ \alpha\ (\eta\ x) & := & x \\
\partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i))
\end{array}$$

For any numeral n:

$$\forall \alpha.\ n = \partial\ (\eta\ n)\ \alpha$$

$$\forall \alpha.\ \alpha\ n = \partial\ (\gamma\ (\eta\ n))\ \alpha$$



II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]^{\alpha}_{\epsilon}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

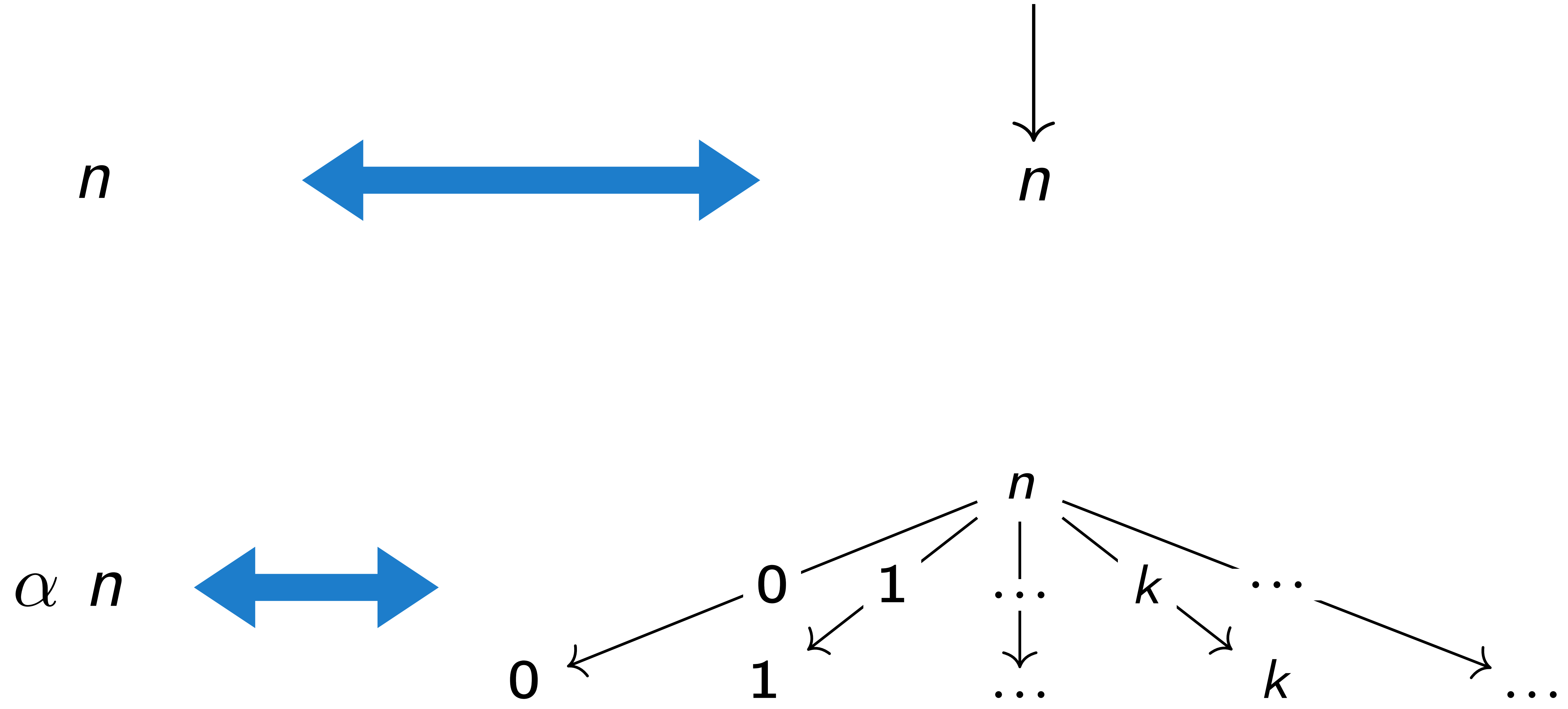
$$\begin{array}{lll}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := & \\
\mid \quad \eta : A \to \mathfrak{D}\ A & & \\
\mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & &
\end{array}$$

$$\begin{array}{lll}
\partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\
\partial\ \alpha\ (\eta\ x) & := & x \\
\partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i))
\end{array}$$

For any numeral n:

$$\forall \alpha.\ n = \partial\ (\eta\ n)\ \alpha$$

$$\forall \alpha.\ \alpha\ n = \partial\ (\gamma\ (\eta\ n))\ \alpha$$



II. The case of System T

What about α ?

$$[\![\mathsf{N} \to \mathsf{N}]\!]_{\epsilon}^{\alpha}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{ll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\ \quad |\quad \eta : A \to \mathfrak{D}\ A & \\ \quad |\quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & \end{array}$$

$$\begin{array}{lll} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

We define

$$\delta : \mathbb{N} \to \mathfrak{D}\ \mathbb{N} := \lambda n.\ \beta\ n\ (\lambda k.\ \eta\ k)$$

II. The case of System T

What about α ?

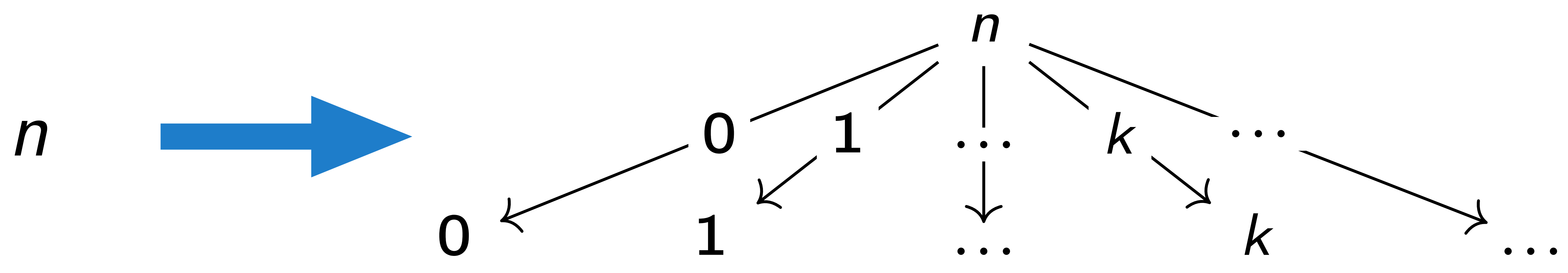
$$[\![\mathsf{N} \to \mathsf{N}]\!]_{\epsilon}^{\alpha}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{ll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := \\ \quad \mid\quad \eta : A \to \mathfrak{D}\ A & \\ \quad \mid\quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & \end{array}$$

$$\begin{array}{lll} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

We define

$$\delta : \mathbb{N} \to \mathfrak{D}\ \mathbb{N} := \lambda n.\ \beta\ n\ (\lambda k.\ \eta\ k)$$



II. The case of System T

What about α ?

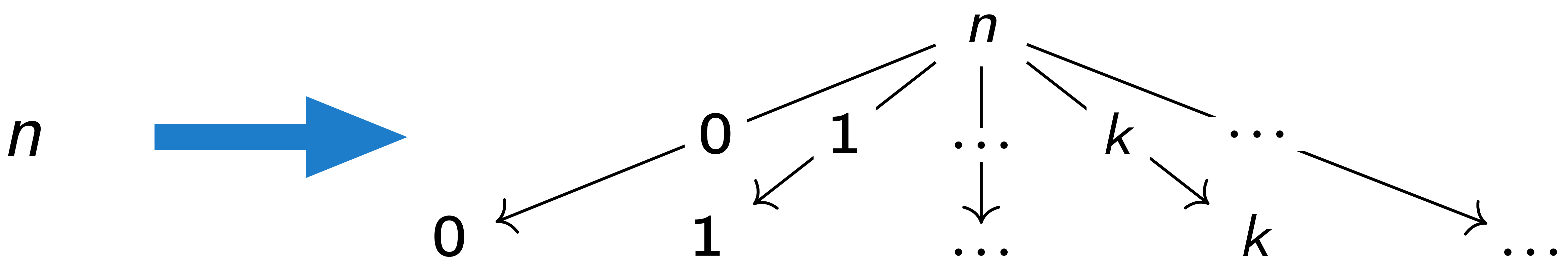
$$[\![\mathrm{N} \to \mathrm{N}]\!]_{\epsilon}^{\alpha}\ \alpha\ \gamma := \forall n\ n_b.\ (n = \partial\ n_b\ \alpha) \longrightarrow \alpha\ n = \partial\ (\gamma\ n_b)\ \alpha$$

$$\begin{array}{lll} \texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := & \\ \mid \quad \eta : A \to \mathfrak{D}\ A & & \\ \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & & \end{array}$$

$$\begin{array}{lll} \partial & : & \Pi\{A : \square\}\,(\alpha : \mathbb{N} \to \mathbb{N})\,(d : \mathfrak{D}\ A).\,A \\ \partial\ \alpha\ (\eta\ x) & := & x \\ \partial\ \alpha\ (\beta\ k\ i) & := & \partial\ \alpha\ (k\ (\alpha\ i)) \end{array}$$

We define

$$\delta : \mathbb{N} \to \mathfrak{D}\ \mathbb{N} := \lambda n.\ \beta\ n\ (\lambda k.\ \eta\ k)$$



$$\gamma : \mathfrak{D}\ \mathbb{N} \to \mathfrak{D}\ \mathbb{N} := \texttt{bind}\ \delta$$

III. Baclofen Type Theory

The effect of effects

$$A, B, M, N ::= \square_i \mid x \mid M\ N \mid \lambda x : A.\, M \mid \Pi x : A.\, M$$

$$\Gamma, \Delta ::= \cdot \mid \Gamma, x : A$$

$$\frac{}{\vdash \cdot} \qquad \frac{\Gamma \vdash A : \square_i}{\vdash \Gamma, x : A} \qquad \frac{\vdash \Gamma \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\vdash \Gamma \qquad i < j}{\Gamma \vdash \square_i : \square_j}$$

$$\frac{\Gamma \vdash A : \square_i \qquad \Gamma \vdash M : B}{\Gamma, x : A \vdash M : B} \qquad \frac{\Gamma \vdash A : \square_i \qquad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Pi x : A.\, B : \square_{\max(i,j)}}$$

$$\frac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B\{x := N\}}$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B : \square_i}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : \square_i \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash M : B}$$

III. Baclofen Type Theory

The effect of effects

$$\texttt{Inductive}\ \mathbb{B} := \mathsf{true} : \mathbb{B} \mid \mathsf{false} : \mathbb{B}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{B} : \square_i} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{true} : \mathbb{B}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{false} : \mathbb{B}}$$

$$\frac{\Gamma \vdash P : \mathbb{B} \to \square_i \qquad \Gamma \vdash t_{\mathsf{true}} : P\ \mathsf{true} \qquad \Gamma \vdash t_{\mathsf{false}} : P\ \mathsf{false}}{\Gamma \vdash \mathbb{B}\_\mathsf{rect}\ P\ \ t_{\mathsf{true}}\ \ t_{\mathsf{false}} : \Pi b : \mathbb{B}.\, P\ \ b}$$

$$\mathbb{B}\_\mathsf{rect}\ P\ \ t_{\mathsf{true}}\ \ t_{\mathsf{false}}\ \mathsf{true} \equiv t_{\mathsf{true}}$$

$$\mathbb{B}\_\mathsf{rect}\ P\ \ t_{\mathsf{true}}\ \ t_{\mathsf{false}}\ \mathsf{false} \equiv t_{\mathsf{false}}$$

III. Baclofen Type Theory

The effect of effects

$$\texttt{Inductive}\ \mathbb{N} := \mathsf{O} : \mathbb{N} \mid \mathsf{S} : \mathbb{N} \to \mathbb{N}$$

$$\dfrac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i} \qquad \dfrac{\Gamma \vdash}{\Gamma \vdash \mathsf{O} : \mathbb{N}} \qquad \dfrac{\Gamma \vdash}{\Gamma \vdash \mathsf{S} : \mathbb{N} \to \mathbb{N}}$$

$$\dfrac{\Gamma \vdash P : \mathbb{N} \to \square_i \qquad \Gamma \vdash t_{\mathsf{O}} : P\ \mathsf{O} \qquad \Gamma \vdash t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n \to P\ (\mathsf{S}\ n)}{\Gamma \vdash \mathbb{N}\_\mathsf{rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n}$$

$$\mathbb{N}\_\mathsf{rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ \mathsf{O} \equiv t_{\mathsf{O}}$$

$$\mathbb{N}\_\mathsf{rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ (\mathsf{S}\ n) \equiv t_{\mathsf{S}}\ n\ (\mathbb{N}\_\mathsf{ind}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ n)$$

III. Baclofen Type Theory

The effect of effects

$$\texttt{Inductive}\ \mathbb{N} := \mathsf{O} : \mathbb{N} \mid \mathsf{S} : \mathbb{N} \to \mathbb{N}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{O} : \mathbb{N}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{S} : \mathbb{N} \to \mathbb{N}}$$

$$\frac{\Gamma \vdash P : \mathbb{N} \to \square_i \qquad \Gamma \vdash t_{\mathsf{O}} : P\ \mathsf{O} \qquad \Gamma \vdash t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n \to P\ (\mathsf{S}\ n)}{\Gamma \vdash \mathbb{N}\text{\_rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n}$$

$$\mathbb{N}\text{\_rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ \mathsf{O} \equiv t_{\mathsf{O}}$$

$$\mathbb{N}\text{\_rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ (\mathsf{S}\ n) \equiv t_{\mathsf{S}}\ n\ (\mathbb{N}\text{\_ind}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}}\ n)$$

$$\begin{array}{lcl}
\texttt{tuple} & : & \Pi\{A : \square\}\,(n : \mathbb{N}).\,\square \\
\texttt{tuple}\ A\ \mathsf{O} & := & \texttt{unit} \\
\texttt{tuple}\ A\ (\mathsf{S}\ k) & := & A\ \times\ (\texttt{tuple}\ A\ k)
\end{array}$$

$$\texttt{tuple}(A : \square)(n : \mathbb{N}) : \square := \mathbb{N}\text{\_rect}\ (\lambda m.\ \square)\ \texttt{unit}\ (\lambda k\ K.\ A \times K)\ n$$

III. Baclofen Type Theory

The effect of effects

$$\texttt{Inductive}\ \mathbb{N} := \mathsf{O} : \mathbb{N} \mid \mathsf{S} : \mathbb{N} \to \mathbb{N}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathbb{N} : \square_i} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{O} : \mathbb{N}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{S} : \mathbb{N} \to \mathbb{N}}$$

$$\frac{\Gamma \vdash P : \mathbb{N} \to \square_i \qquad \Gamma \vdash t_{\mathsf{O}} : P\ \mathsf{O} \qquad \Gamma \vdash t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n \to P\ (\mathsf{S}\ n)}{\Gamma \vdash \mathbb{N}\_\mathsf{rect}\ P\ t_{\mathsf{O}}\ t_{\mathsf{S}} : \Pi n : \mathbb{N}.\, P\ n}$$

III. Baclofen Type Theory

The effect of effects

Theorem

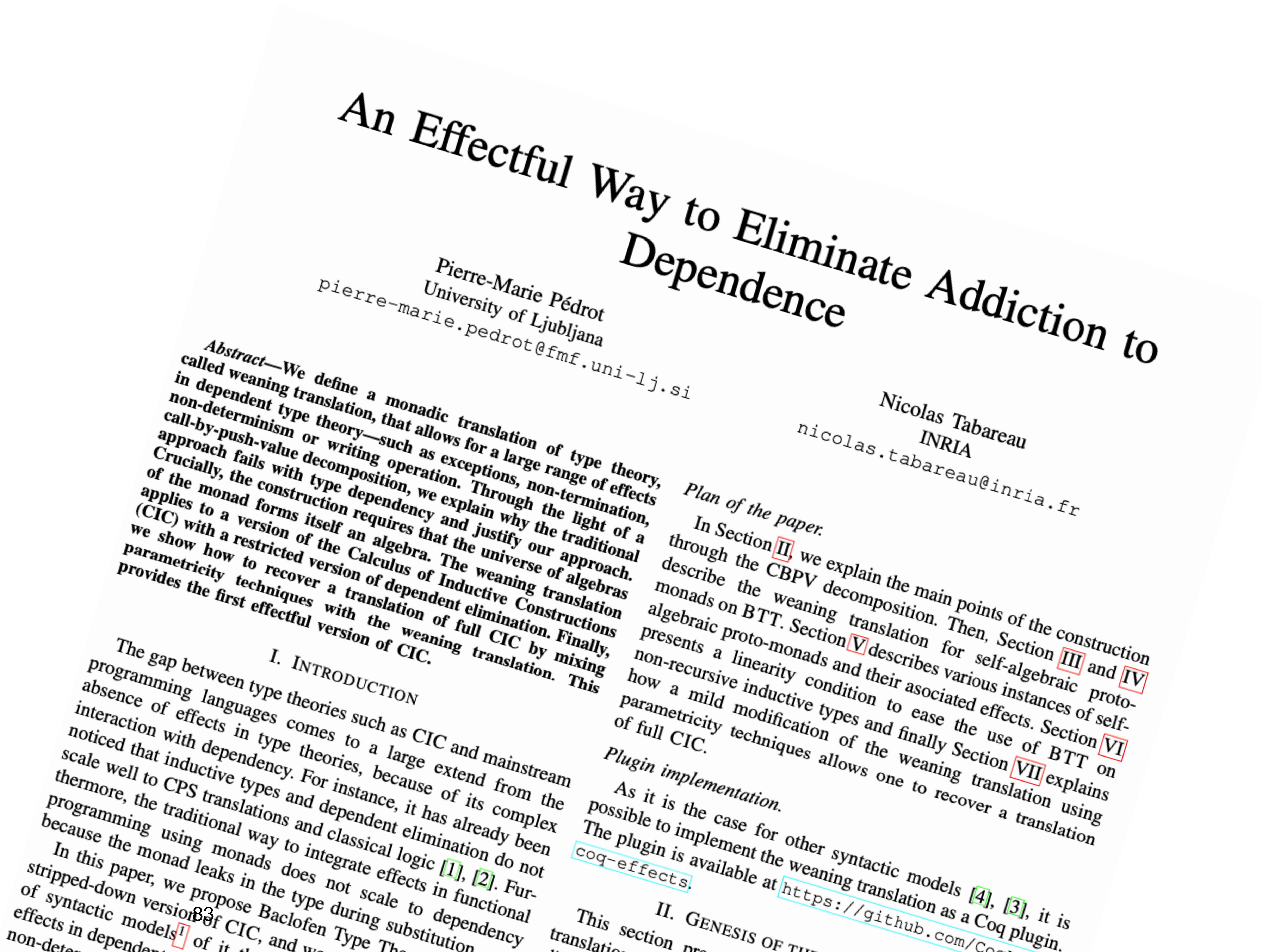
A dependent type theory that features

1. Dependent elimination
2. Substitution
3. An observable effect

is inconsistent.

III. Baclofen Type Theory

BTT



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th
stripped-down version of CIC, and w
of syntactic models[1] of it th
effects in dependen
non-dete

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

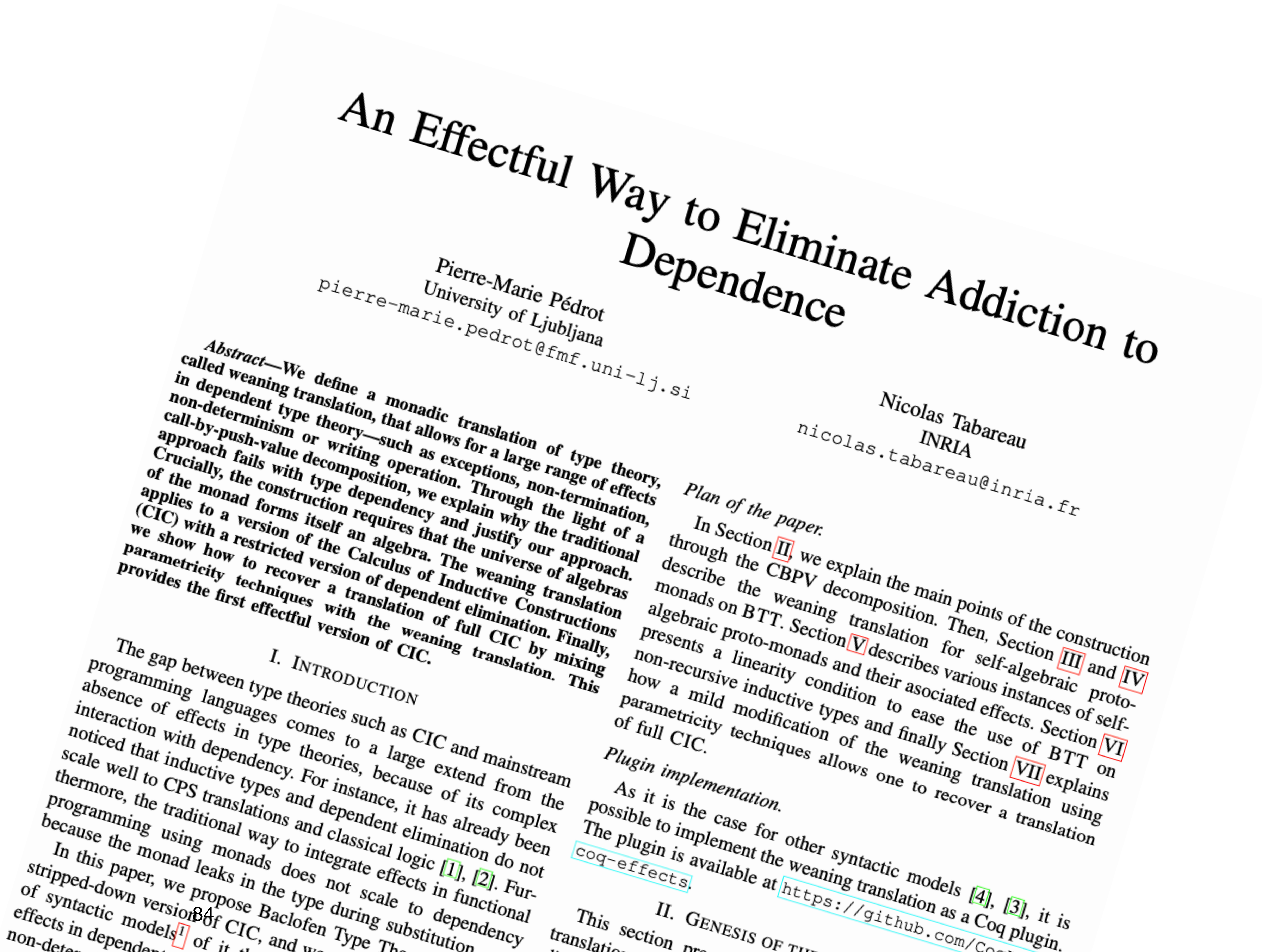
As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq coq-effects.

II. Genesis of th

This section pr
translatio

III. Baclofen Type Theory

BTT: bad pun



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th
stripped-down version of CIC, and w
of syntactic models[1] of it th
effects in dependen
non-deter

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq
coq-effects.

II. Genesis of th

This section pr
translatio

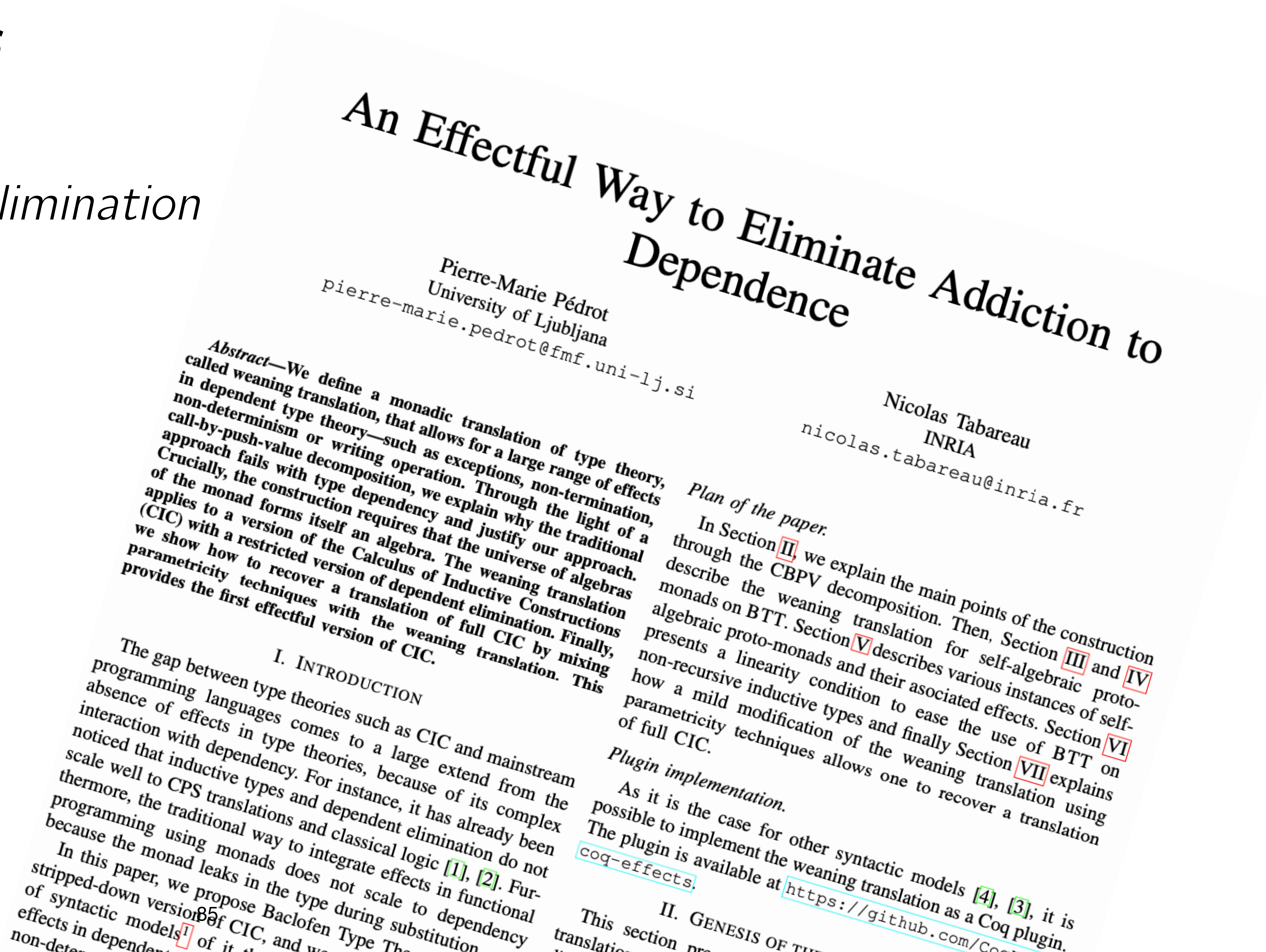
III. Baclofen Type Theory

But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th
stripped-down version of CIC, and w
of syntactic models[1] of it th
effects in dependen
non-deter

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq
coq-effects.

II. Genesis of th

This section pr
translatio

III. Baclofen Type Theory

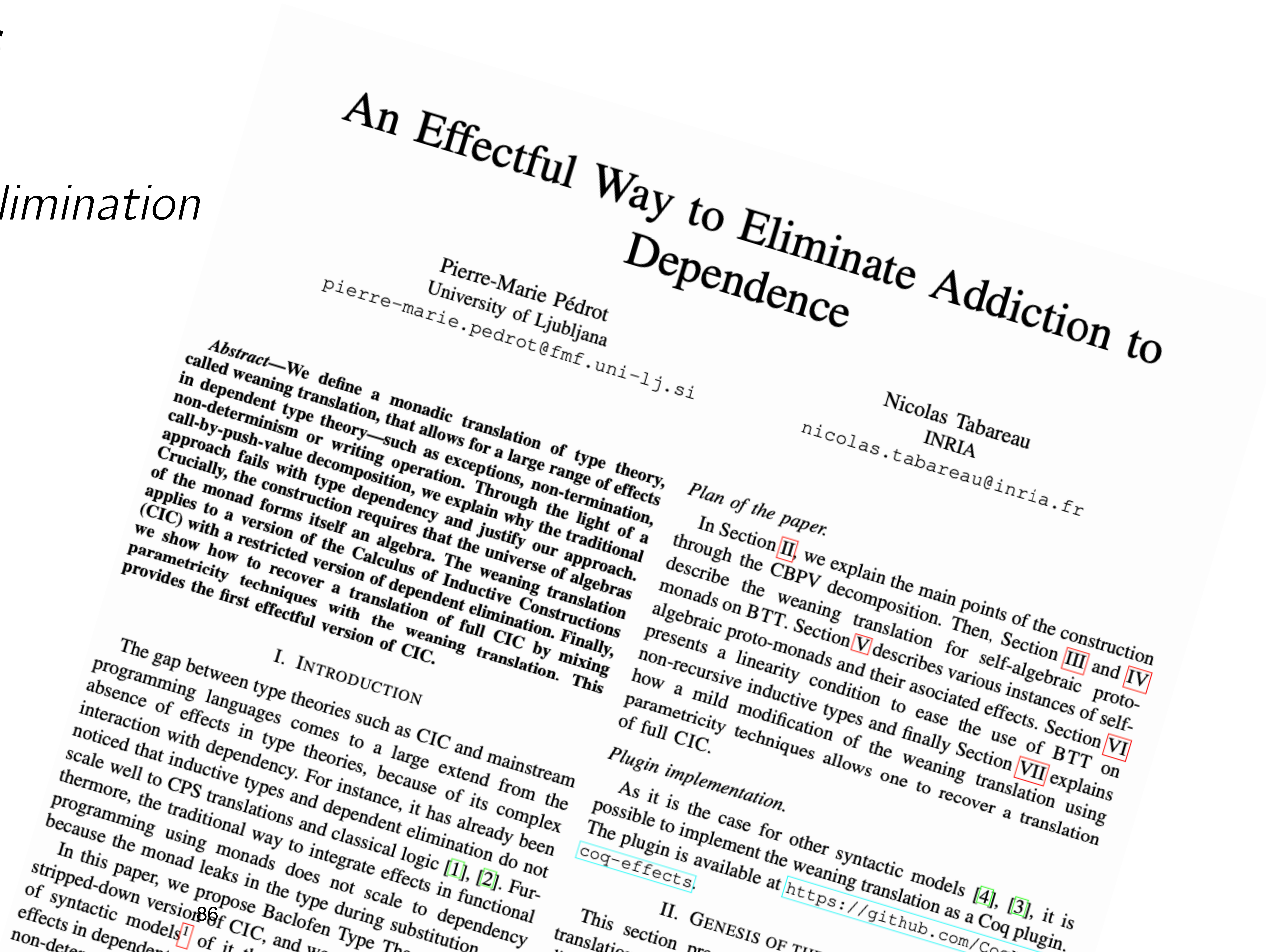
But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects

Exceptions



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th
stripped-down version of CIC, and w
of syntactic models[1] of it th
effects in dependen
non-dete

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq
coq-effects.

II. Genesis of th

This section pr
translatio

III. Baclofen Type Theory

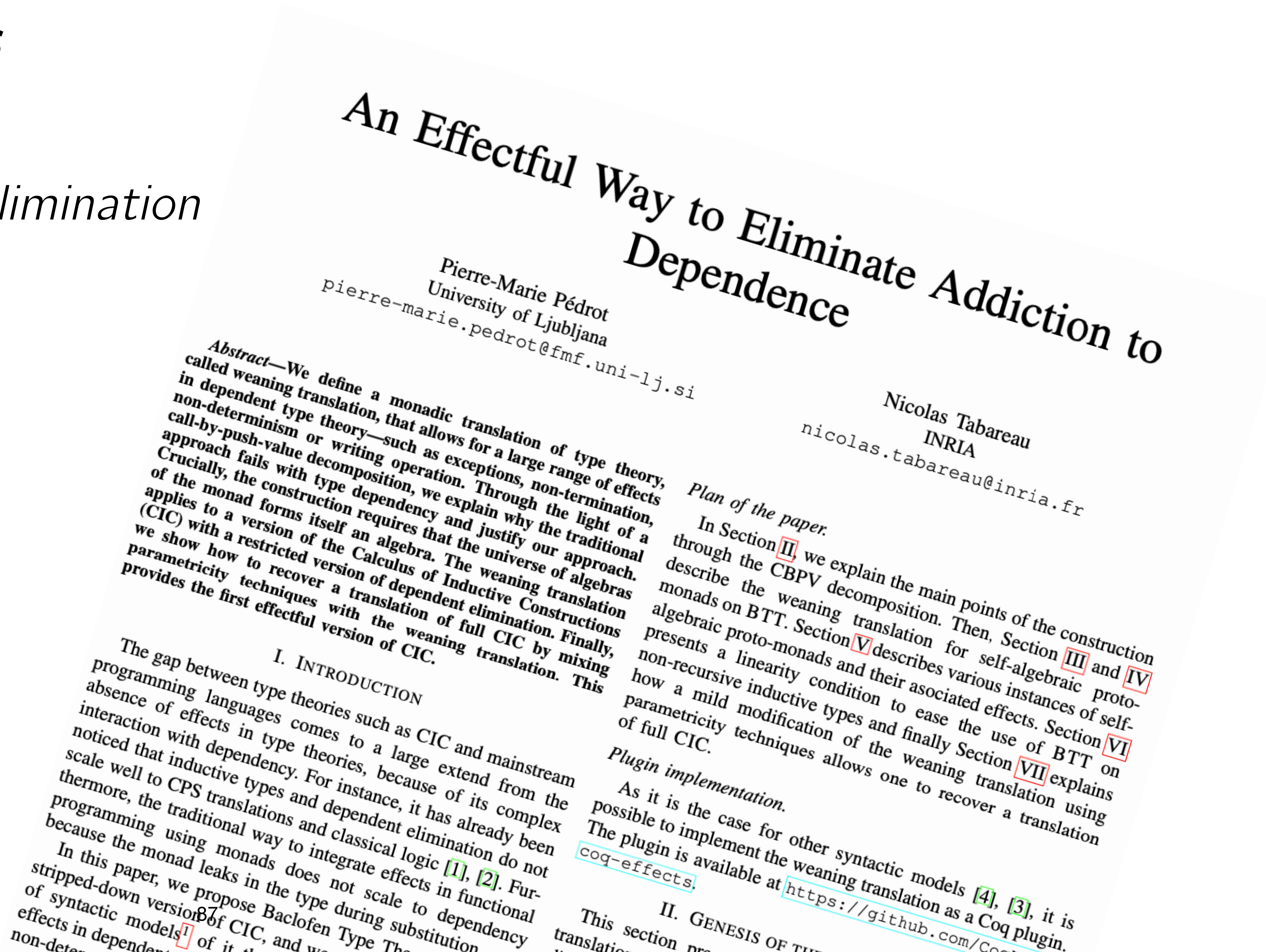
But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects

- *Exceptions*
- *Non-termination*



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th

stripped-down version of CIC, and w

of syntactic models[1] of it th

effects in dependen

non-dete

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq coq-effects.

II. Genesis of th

This section pro

translatio

III. Baclofen Type Theory

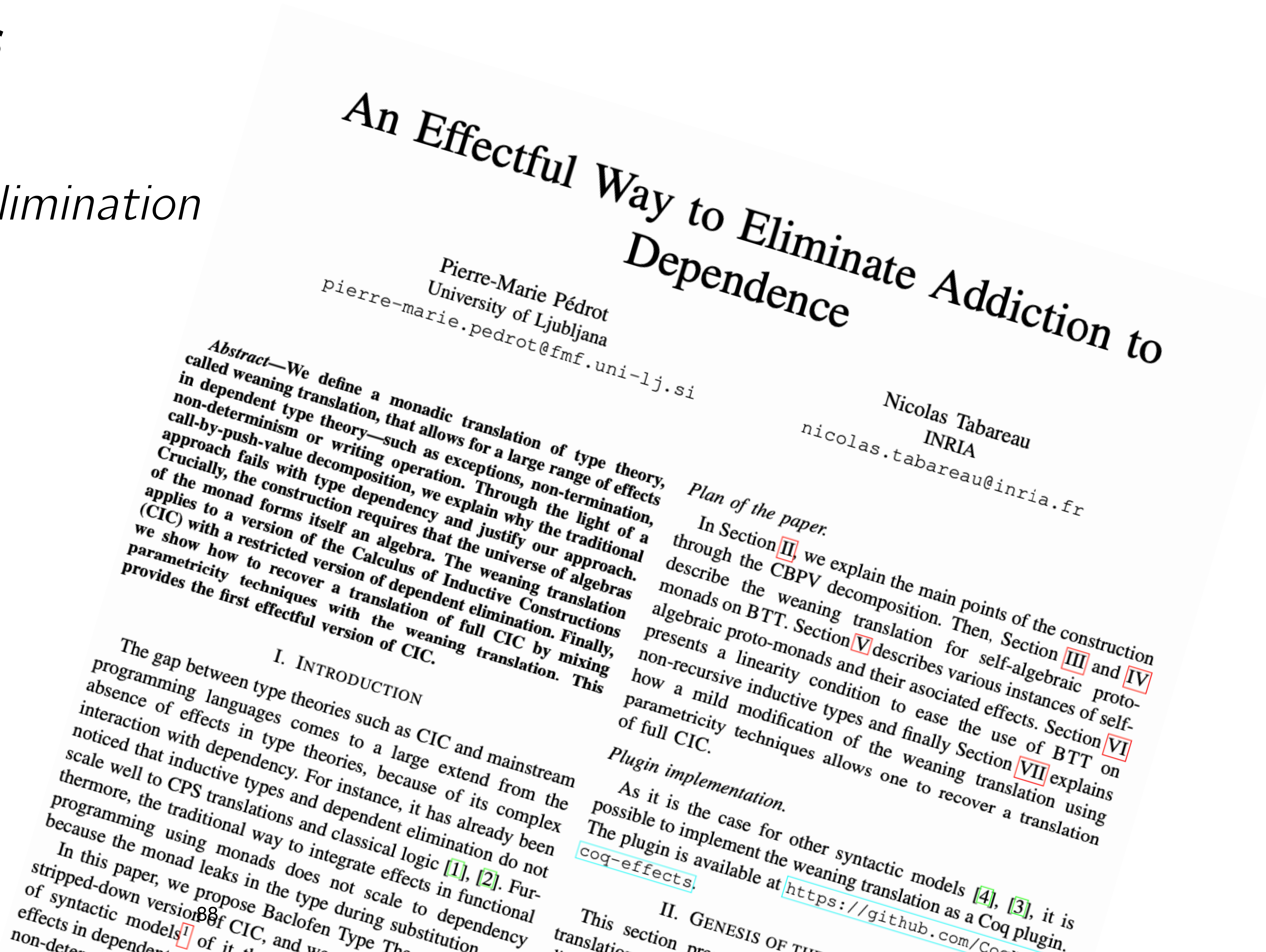
But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects

- *Exceptions*
- *Non-termination*
- *Non-determinism*



An Effectful Way to Eliminate Addiction to Dependence

Pierre-Marie Pédrot
University of Ljubljana
pierre-marie.pedrot@fmf.uni-lj.si

Nicolas Tabareau
INRIA
nicolas.tabareau@inria.fr

***Abstract*—We define a monadic translation of type theory, called weaning translation, that allows for a large range of effects in dependent type theory—such as exceptions, non-termination, non-determinism or writing operation. Through the light of a call-by-push-value decomposition, we explain why the traditional approach fails with type dependency and justify our approach. Crucially, the construction requires that the universe of algebras of the monad forms itself an algebra. The weaning translation applies to a version of the Calculus of Inductive Constructions (CIC) with a restricted version of dependent elimination. Finally, we show how to recover a translation of full CIC by mixing parametricity techniques with the weaning translation. This provides the first effectful version of CIC.**

I. Introduction

The gap between type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1], [2]. Furthermore, the traditional way to integrate effects in functional programming using monads does not scale to dependency because the monad leaks in the type during substitution

In this paper, we propose Baclofen Type Th
stripped-down version of CIC, and w
of syntactic models[1] of it t
effects in dependen
non-deter

Plan of the paper.

In Section II, we explain the main points of the construction through the CBPV decomposition. Then, Section III and IV describe the weaning translation for self-algebraic proto-monads on BTT. Section V describes various instances of self-algebraic proto-monads and their asociated effects. Section VI presents a linearity condition to ease the use of BTT on non-recursive inductive types and finally Section VII explains how a mild modification of the weaning translation using parametricity techniques allows one to recover a translation of full CIC.

Plugin implementation.

As it is the case for other syntactic models [4], [3], it is possible to implement the weaning translation as a Coq plugin. The plugin is available at https://github.com/Coq
coq-effects.

II. Genesis of th

This section pr
translatio

III. Baclofen Type Theory

But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\text{\_rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\ P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\text{\_rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\ \mathbb{B}\text{\_store}\ P\ b}$$

$\mathbb{B}\text{\_store}\ P\ \texttt{true} \equiv P\ \texttt{true}$

$\mathbb{B}\text{\_store}\ P\ \texttt{false} \equiv P\ \texttt{false}$

$\mathbb{B}\text{\_store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

III. Baclofen Type Theory

But good theory nonetheless

BTT = Dependent Type Theory

with restricted dependent elimination

to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\_\mathsf{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\_\mathsf{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, \mathbb{B}\_\mathsf{store}\ P\ b}$$

$\mathbb{B}\_\mathsf{store}\ P\ \texttt{true} \equiv P\ \texttt{true}$

$\mathbb{B}\_\mathsf{store}\ P\ \texttt{false} \equiv P\ \texttt{false}$

$\mathbb{B}\_\mathsf{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

$$\texttt{tuple}(A : \square)(n : \mathbb{N}) : \square := \mathbb{N}\_\mathsf{rect}\ (\lambda m.\ \square)\ \texttt{unit}\ (\lambda k\ K.\ A \times K)\ n$$

$$\begin{array}{lcl}
\texttt{tuple} & : & \Pi\{A : \square\}\,(n : \mathbb{N}).\,\square \\
\texttt{tuple}\ A\ \mathsf{O} & := & \texttt{unit} \\
\texttt{tuple}\ A\ (\mathsf{S}\ k) & := & A\ \times\ (\texttt{tuple}\ A\ k) \\
\texttt{tuple}\ A\ \beta & := & \text{underspecified}
\end{array}$$

IV. The Dialogue Model of BTT

Big Tree Theory

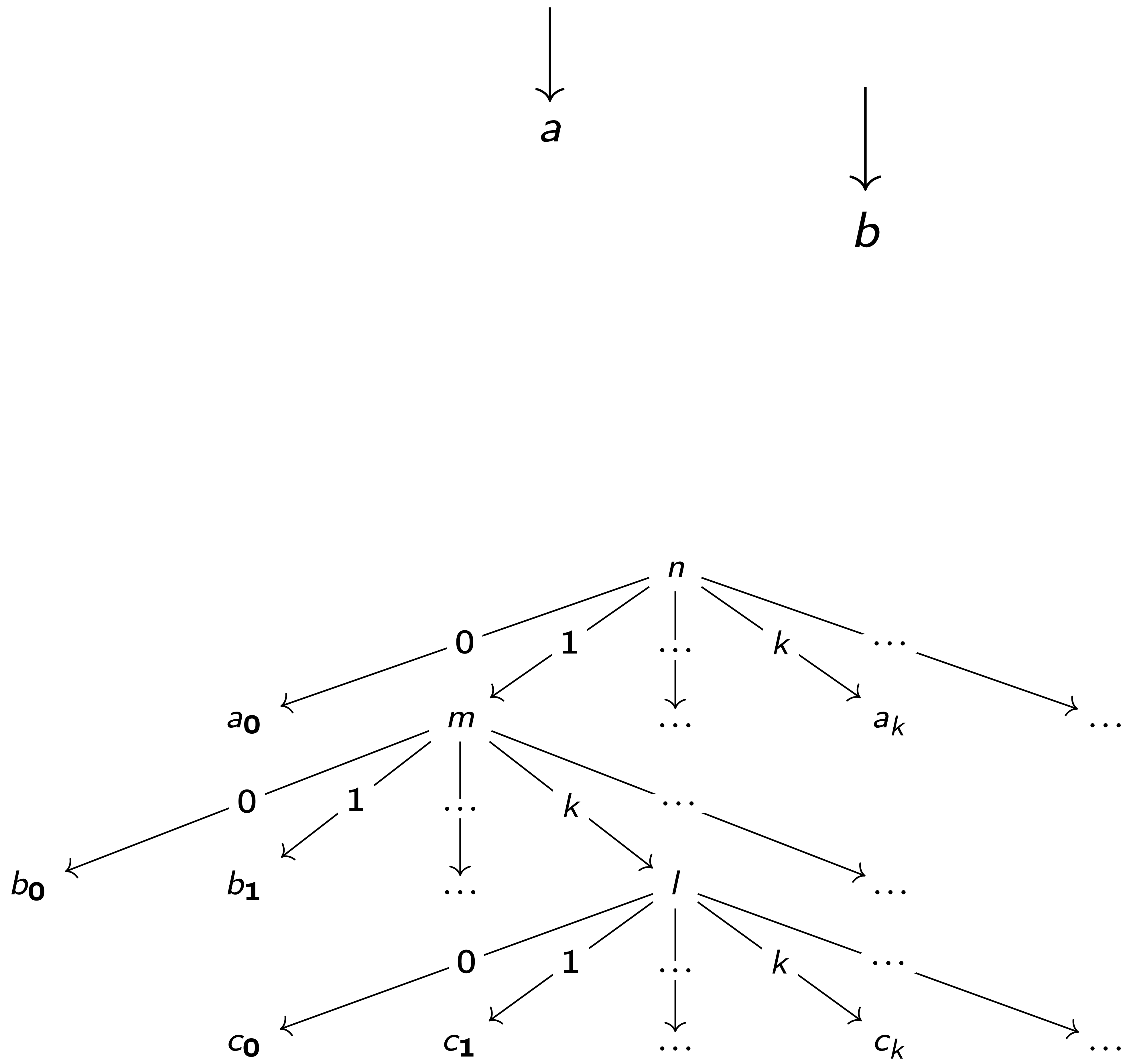
IV. The Dialogue Model of BTT

Talking trees

We consider the following ***Dialogue*** *operator :*

$$
\begin{array}{lll}
\texttt{Inductive}\ \mathfrak{D}\ (A : \square) : \square & := & \\
\quad | \quad \eta : A \to \mathfrak{D}\ A & & \\
\quad | \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A. & &
\end{array}
$$

$(\mathfrak{D},\ \eta,\ \texttt{bind})$ is a *"moral"* monad

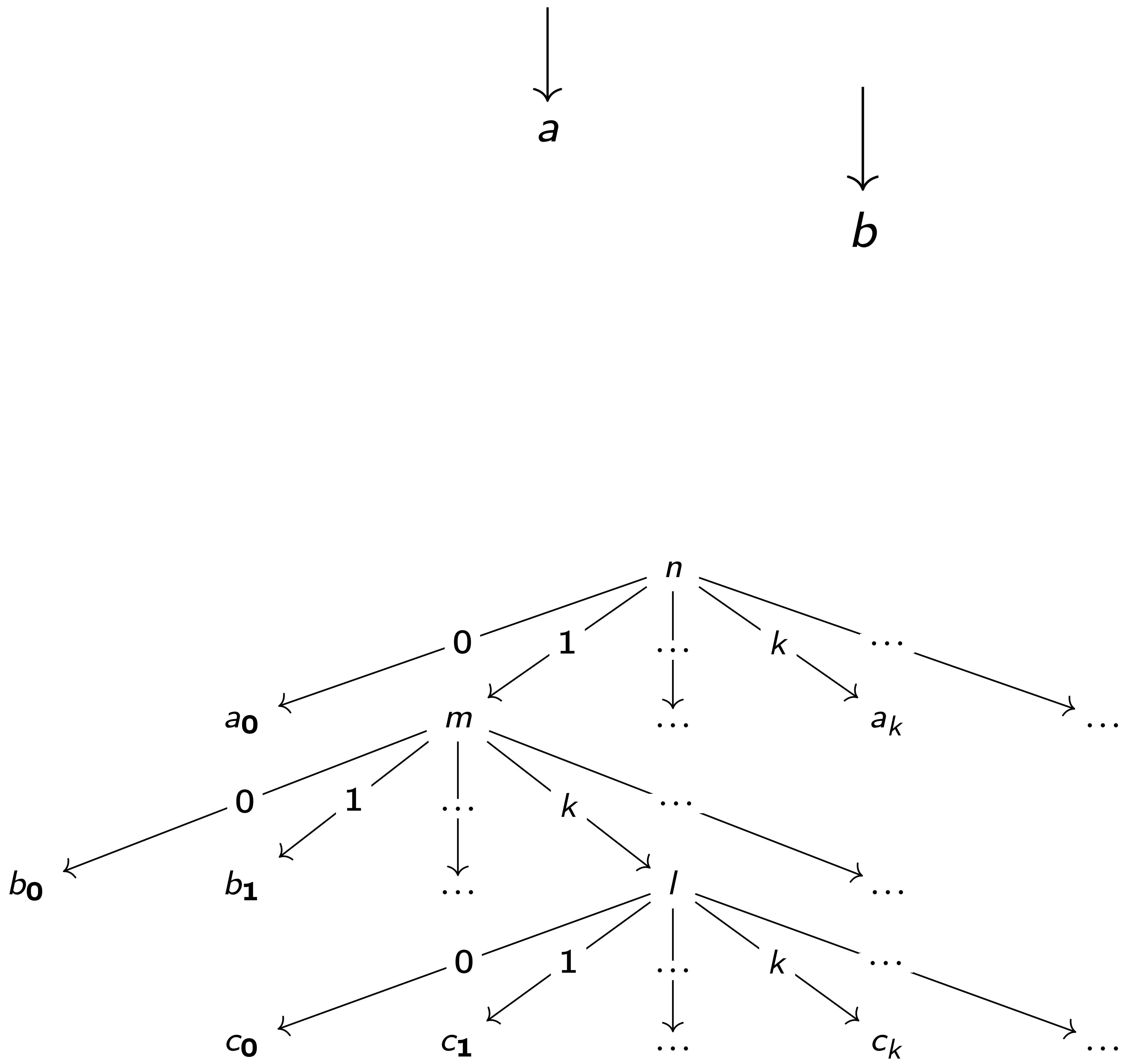


IV. The Dialogue Model of BTT

Talking trees

We consider the following ***Dialogue*** *operator :*

$$
\begin{aligned}
&\texttt{Inductive}\ \mathfrak{D}\ (A:\square):\square \qquad := \\
&\quad \mid \quad \eta : A \to \mathfrak{D}\ A \\
&\quad \mid \quad \beta : (\mathbb{N} \to \mathfrak{D}\ A) \to \mathbb{N} \to \mathfrak{D}\ A.
\end{aligned}
$$



($\mathfrak{D}$, η, $\texttt{bind}$) is a ''*moral*'' monad

We can build a BTT Model where types are interpreted as ''moral'' algebras of $\mathfrak{D}$

IV. The Dialogue Model of BTT

Moral-blablabla

We define the type of ''moral'' algebras of the **Dialogue** *''moral'' monad:*

$$\boxed{\square}^b \quad \approx \quad \Sigma(A : \square).\ \mathsf{isAlg}_{\mathfrak{D}}(A)$$

IV. The Dialogue Model of BTT

Moral-blablabla

We define the type of ''moral'' algebras of the **Dialogue** *''moral'' monad:*

$$\begin{aligned} \mathbb{\square}^b &\approx \Sigma(A : \square).\ \mathsf{isAlg}_{\mathfrak{D}}(A) \\ \mathbb{\square}^b &:= \Sigma(A : \square).\ \Pi(i : \mathbb{N}).\ (\mathbb{N} \to A) \to A \end{aligned}$$

IV. The Dialogue Model of BTT

Moral-blablabla

*We define the type of ''moral'' algebras of the **Dialogue** ''moral'' monad:*

$$\boxed{\square}^{b} \quad \approx \quad \Sigma(A : \square).\ \mathrm{isAlg}_{\mathfrak{D}}(A)$$

$$\boxed{\square}^{b} \quad := \quad \Sigma(A : \square).\ \Pi(i : \mathbb{N}).\,(\mathbb{N} \to A) \to A$$

*We call such types **Branching** types*

IV. The Dialogue Model of BTT

The Branching translation

$$\Gamma \vdash t : A \qquad \xrightarrow[\textit{Translation}]{\textit{Branching}} \qquad [\![\Gamma]\!]_b \vdash [t]_b : [\![A]\!]_b$$

$$[\![\Box]\!]_b \qquad := \qquad \Box^b$$

IV. The Dialogue Model of BTT

The Branching translation

Inductive $\mathbb{B}_b$:=
| $\mathit{true}_b : \mathbb{B}_b$
| $\mathit{false}_b : \mathbb{B}_b$
| $\beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b$.

Inductive $\mathbb{N}_b$:=
| $0_b : \mathbb{N}_b$
| $S_b : \mathbb{N}_b \to \mathbb{N}_b$
| $\beta_{\mathbb{N}_b} : (\mathbb{N} \to \mathbb{N}_b) \to \mathbb{N} \to \mathbb{N}_b$.

IV. The Dialogue Model of BTT

The Branching translation

$$[A]_b \quad := \quad (\llbracket A \rrbracket_b, \beta_A)$$

IV. The Dialogue Model of BTT

The Branching translation

$$
\begin{aligned}
[A]_b &:= (\llbracket A \rrbracket_b, \beta_A) \\
\\
\llbracket \mathbb{N} \rrbracket_b &:= \mathbb{N}_b \\
\beta_{\mathbb{N}} &:= \beta_{\mathbb{N}_b}
\end{aligned}
$$

IV. The Dialogue Model of BTT

The Branching translation

$$
\begin{aligned}
[A]_b &:= (\llbracket A \rrbracket_b, \beta_A) \\[1em]
\llbracket \mathbb{N} \rrbracket_b &:= \mathbb{N}_b \\
\beta_{\mathbb{N}} &:= \beta_{\mathbb{N}_b} \\[1em]
\llbracket \square \rrbracket_b &:= \mathbb{\square}^b \\
\beta_{\square} &:= \lambda(i : \mathbb{N})\,(k : \mathbb{N} \to \mathbb{\square}^b).\,\mho_b
\end{aligned}
$$

IV. The Dialogue Model of BTT

The Branching translation

$$
\begin{array}{lcl}
[A]_b & := & (\llbracket A \rrbracket_b, \beta_A) \\
\\
\llbracket \mathbb{N} \rrbracket_b & := & \mathbb{N}_b \\
\beta_{\mathbb{N}} & := & \beta_{\mathbb{N}_b} \\
\\
\llbracket \Box \rrbracket_b & := & \Box^b \\
\beta_{\Box} & := & \lambda(i : \mathbb{N})\,(k : \mathbb{N} \to \Box^b).\, \mho_b \\
\\
\llbracket \Pi x : A.\, B \rrbracket_b & := & \Pi x_b : \llbracket A \rrbracket_b.\, \llbracket B \rrbracket_b \\
\beta_{\Pi x : A.\, B} & := & \lambda(i : \mathbb{N})\,(k : \mathbb{N} \to \Pi x : \llbracket A \rrbracket_b.\, \llbracket B \rrbracket_b)\,(x : \llbracket A \rrbracket_b). \\
 & & \qquad \beta_B\ i\ (\lambda n : \mathbb{N}.\, k\ n\ x)
\end{array}
$$

IV. The Dialogue Model of BTT

The Branching translation

$$
\begin{array}{lcl}
[x]_b & := & x_b \\[2em]
[\lambda x : A.\, M]_b & := & \lambda x_b : [\![A]\!]_b .\, [M]_b \\[2em]
[M\ N]_b & := & [M]_b\ [N]_b \\[2em]
[\![\cdot]\!]_b & := & \cdot \\
[\![\Gamma, x : A]\!]_b & := & [\![\Gamma]\!]_b, x_b : [\![A]\!]_b
\end{array}
$$

V. The full model

3 syntactic translations for the price of 1

V. The full model

A sense of déjà-vu

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash t : \mathbb{N}$$

Black box

$$\vdash d : \mathbb{N}_b$$

Explicit calls

V. The full model

A sense of déjà-vu

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash t : \mathbb{N} \qquad\qquad \vdash d : \mathbb{N}_b$$

Black box ↔ *Explicit calls*

V. The full model

A sense of déjà-vu

$\alpha : \mathbb{N} \to \mathbb{N} \vdash t : \mathbb{N}$ $\vdash d : \mathbb{N}_b$

Black box ⟷ *Explicit calls*

Binary Parametricity

V. The full model

A sense of déjà-vu

Given $\alpha : \mathbb{N} \to \mathbb{N}$, $\Gamma \vdash t : A$ will be translated as :

$$[\![\Gamma]\!]_a^\alpha \vdash [t]_a^\alpha : [\![A]\!]_a^\alpha$$ *Axiom translation*

$$[\![\Gamma]\!]_b \vdash [t]_b : [\![A]\!]_b$$ *Branching translation*

$$[\![\Gamma]\!]_\epsilon^\alpha \vdash [t]_\epsilon^\alpha : [\![A]\!]_\epsilon^\alpha \; [t]_a^\alpha \; [t]_b$$ *Parametricity translation*

V. The full model

The example of booleans

$([\![\mathbb{B}]\!]^{\alpha}_{a},\ [\![\mathbb{B}]\!]_{b},\ [\![\mathbb{B}]\!]^{\alpha}_{\epsilon}) \equiv (\mathbb{B},\ \mathbb{B}_{b},\ \mathbb{B}^{\alpha}_{\epsilon})$ *where* :

V. The full model

The example of booleans

$$(\llbracket \mathbb{B} \rrbracket^{\alpha}_{a},\ \llbracket \mathbb{B} \rrbracket_{b},\ \llbracket \mathbb{B} \rrbracket^{\alpha}_{\epsilon}) \equiv (\mathbb{B},\ \mathbb{B}_{b},\ \mathbb{B}^{\alpha}_{\epsilon})\ \textit{where :}$$

$\texttt{Inductive}\ \mathbb{B} :=$
$|\ \mathit{true} : \mathbb{B}$
$|\ \mathit{false} : \mathbb{B}.$

V. The full model

The example of booleans

$$([\![\mathbb{B}]\!]_a^\alpha,\ [\![\mathbb{B}]\!]_b,\ [\![\mathbb{B}]\!]_\epsilon^\alpha) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}_\epsilon^\alpha)\ \textit{where}:$$

`Inductive` $\mathbb{B}$:=
| *true* : $\mathbb{B}$
| *false* : $\mathbb{B}$.

`Inductive` $\mathbb{B}_b$:=
| $\mathit{true}_b : \mathbb{B}_b$
| $\mathit{false}_b : \mathbb{B}_b$
| $\beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b$.

V. The full model

The example of booleans

$$([\![\mathbb{B}]\!]_a^{\alpha},\ [\![\mathbb{B}]\!]_b,\ [\![\mathbb{B}]\!]_{\epsilon}^{\alpha}) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}_{\epsilon}^{\alpha})\ \textit{where} :$$

$\texttt{Inductive}\ \mathbb{B} :=$
$\mid\ \mathit{true} : \mathbb{B}$
$\mid\ \mathit{false} : \mathbb{B}.$

$\texttt{Inductive}\ \mathbb{B}_b :=$
$\mid\ \mathit{true}_b : \mathbb{B}_b$
$\mid\ \mathit{false}_b : \mathbb{B}_b$
$\mid\ \beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b.$

$\texttt{Inductive}\ \mathbb{B}_{\epsilon}^{\alpha} : \mathbb{B} \to \mathbb{B}_b \to \square_i :=$
$\mid\ \mathit{true}_{\epsilon}^{\alpha} : \mathbb{B}_{\epsilon}^{\alpha}\ \mathit{true}\ \mathit{true}_b$
$\mid\ \mathit{false}_{\epsilon}^{\alpha} : \mathbb{B}_{\epsilon}^{\alpha}\ \mathit{false}\ \mathit{false}_b$
$\mid\ \beta_{\mathbb{B}_{\epsilon}^{\alpha}} : \forall\ (b_a : \mathbb{B})$
$\quad (f : \mathbb{N} \to \mathbb{B}_b)$
$\quad (n : \mathbb{N})$
$\quad (b_{\epsilon} : \mathbb{B}_{\epsilon}^{\alpha}\ b_a\ (f\ (\alpha\ n))),$
$\quad \mathbb{B}_{\epsilon}^{\alpha}\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).$

V. The full model

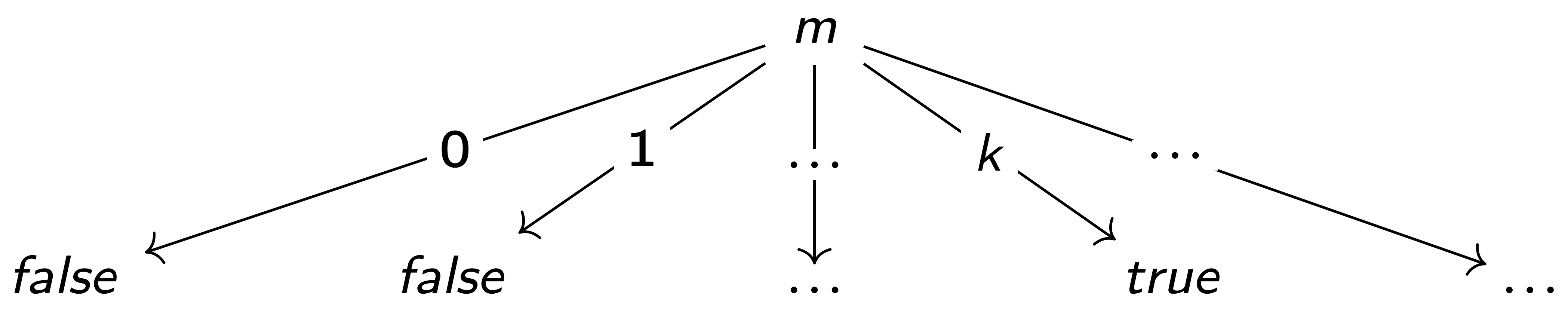
The example of booleans

$$([\![\mathbb{B}]\!]^{\alpha}_{a},\ [\![\mathbb{B}]\!]_{b},\ [\![\mathbb{B}]\!]^{\alpha}_{\epsilon}) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}^{\alpha}_{\epsilon})\ \textit{where}\ :$$

`Inductive` $\mathbb{B} :=$
$\mid \mathit{true} : \mathbb{B}$
$\mid \mathit{false} : \mathbb{B}.$

`Inductive` $\mathbb{B}_b :=$
$\mid \mathit{true}_b : \mathbb{B}_b$
$\mid \mathit{false}_b : \mathbb{B}_b$
$\mid \beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b.$

`Inductive` $\mathbb{B}^{\alpha}_{\epsilon} : \mathbb{B} \to \mathbb{B}_b \to \square_i :=$
$\mid \mathit{true}^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ \mathit{true}\ \mathit{true}_b$
$\mid \mathit{false}^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ \mathit{false}\ \mathit{false}_b$
$\mid \beta_{\mathbb{B}^{\alpha}_{\epsilon}} : \forall\ (b_a : \mathbb{B})$
$\qquad (f : \mathbb{N} \to \mathbb{B}_b)$
$\qquad (n : \mathbb{N})$
$\qquad (b_\epsilon : \mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (f\ (\alpha\ n))),$
$\quad \mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).$



V. The full model

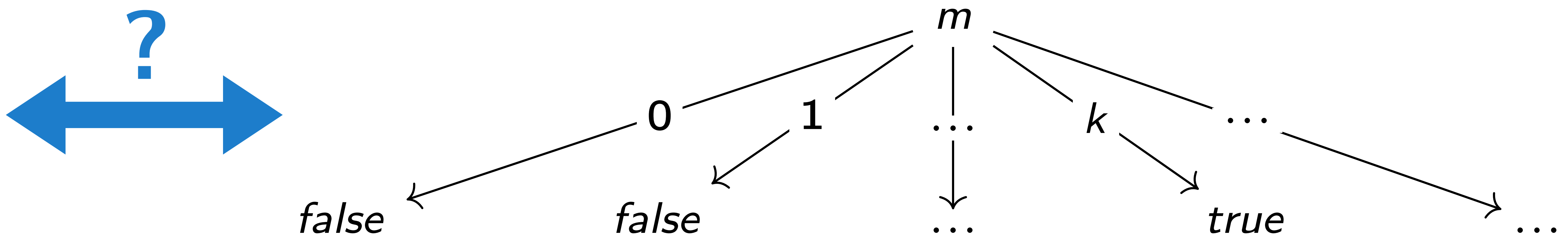
The example of booleans

$$([\![\mathbb{B}]\!]^{\alpha}_{a},\ [\![\mathbb{B}]\!]_{b},\ [\![\mathbb{B}]\!]^{\alpha}_{\epsilon}) \equiv (\mathbb{B},\ \mathbb{B}_{b},\ \mathbb{B}^{\alpha}_{\epsilon})$$ *where* :

`Inductive` $\mathbb{B} :=$
| $true : \mathbb{B}$
| $false : \mathbb{B}.$

`Inductive` $\mathbb{B}_b :=$
| $true_b : \mathbb{B}_b$
| $false_b : \mathbb{B}_b$
| $\beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b.$

`Inductive` $\mathbb{B}^{\alpha}_{\epsilon} : \mathbb{B} \to \mathbb{B}_b \to \square_i :=$
| $true^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ true\ true_b$
| $false^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ false\ false_b$
| $\beta_{\mathbb{B}^{\alpha}_{\epsilon}} : \forall\ (b_a : \mathbb{B})$
$(f : \mathbb{N} \to \mathbb{B}_b)$
$(n : \mathbb{N})$
$(b_\epsilon : \mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (f\ (\alpha\ n))),$
$\mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).$



V. The full model

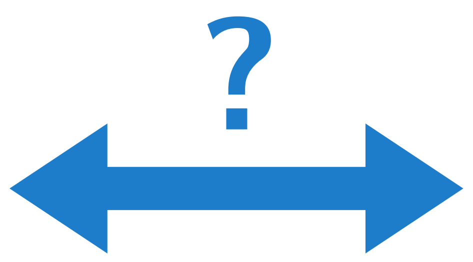
The example of booleans

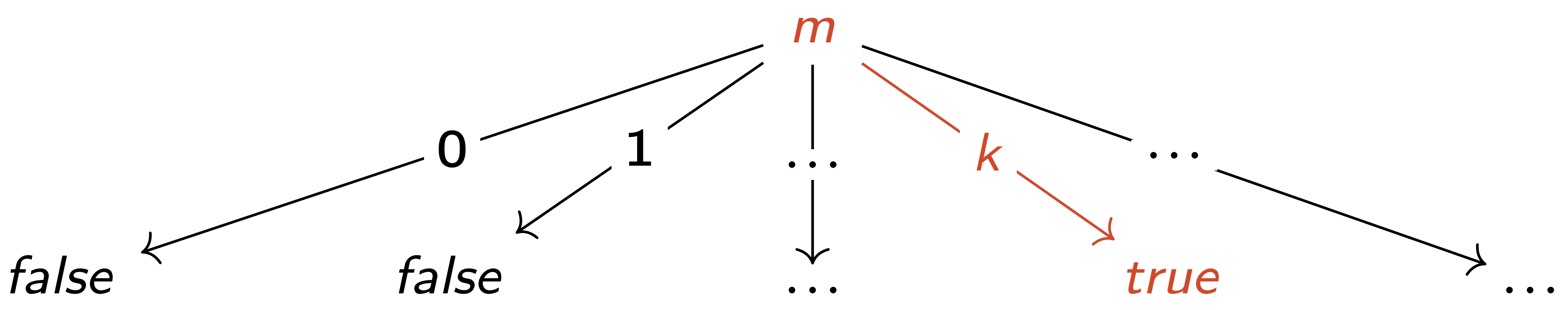
$$(\llbracket\mathbb{B}\rrbracket^{\alpha}_{a},\ \llbracket\mathbb{B}\rrbracket_{b},\ \llbracket\mathbb{B}\rrbracket^{\alpha}_{\epsilon}) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}^{\alpha}_{\epsilon})\ \textit{where :}$$

`Inductive` $\mathbb{B}$:=
| $\mathit{true} : \mathbb{B}$
| $\mathit{false} : \mathbb{B}$.

`Inductive` $\mathbb{B}_b$:=
| $\mathit{true}_b : \mathbb{B}_b$
| $\mathit{false}_b : \mathbb{B}_b$
| $\beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b$.

`Inductive` $\mathbb{B}^{\alpha}_{\epsilon} : \mathbb{B} \to \mathbb{B}_b \to \square_i$:=
| $\mathit{true}^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ \mathit{true}\ \mathit{true}_b$
| $\mathit{false}^{\alpha}_{\epsilon} : \mathbb{B}^{\alpha}_{\epsilon}\ \mathit{false}\ \mathit{false}_b$
| $\beta_{\mathbb{B}^{\alpha}_{\epsilon}} : \forall\ (b_a : \mathbb{B})$
$(f : \mathbb{N} \to \mathbb{B}_b)$
$(n : \mathbb{N})$
$(b_\epsilon : \mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (f\ (\alpha\ n))),$
$\mathbb{B}^{\alpha}_{\epsilon}\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).$





$$\alpha\ m = k$$

V. The full model

The example of booleans

$$([\![\mathbb{B}]\!]_a^\alpha,\ [\![\mathbb{B}]\!]_b,\ [\![\mathbb{B}]\!]_\epsilon^\alpha) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}_\epsilon^\alpha)\ \textit{where} :$$

`Inductive` $\mathbb{B}$:=

| $\mathit{true} : \mathbb{B}$

| $\mathit{false} : \mathbb{B}.$

`Inductive` $\mathbb{B}_b$:=

| $\mathit{true}_b : \mathbb{B}_b$

| $\mathit{false}_b : \mathbb{B}_b$

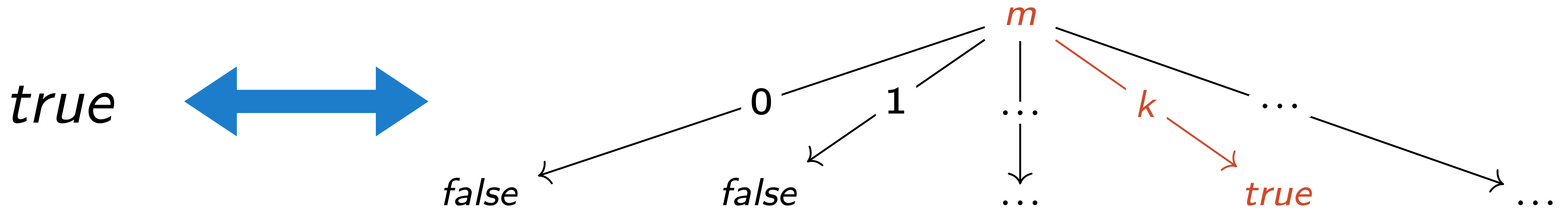
| $\beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b.$

`Inductive` $\mathbb{B}_\epsilon^\alpha : \mathbb{B} \to \mathbb{B}_b \to \square_i$:=

| $\mathit{true}_\epsilon^\alpha : \mathbb{B}_\epsilon^\alpha\ \mathit{true}\ \mathit{true}_b$

| $\mathit{false}_\epsilon^\alpha : \mathbb{B}_\epsilon^\alpha\ \mathit{false}\ \mathit{false}_b$

| $\beta_{\mathbb{B}_\epsilon^\alpha} : \forall\ (b_a : \mathbb{B})$
$(f : \mathbb{N} \to \mathbb{B}_b)$
$(n : \mathbb{N})$
$(b_\epsilon : \mathbb{B}_\epsilon^\alpha\ b_a\ (f\ (\alpha\ n))),$
$\mathbb{B}_\epsilon^\alpha\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).$



$$\alpha\ m = k$$

V. The full model

The example of booleans

$$([\![\mathbb{B}]\!]_a^\alpha,\ [\![\mathbb{B}]\!]_b,\ [\![\mathbb{B}]\!]_\epsilon^\alpha) \equiv (\mathbb{B},\ \mathbb{B}_b,\ \mathbb{B}_\epsilon^\alpha)\ \textit{where} :$$

$$\begin{array}{l}
\texttt{Inductive}\ \mathbb{B} := \\
\mid\ \mathit{true} : \mathbb{B} \\
\mid\ \mathit{false} : \mathbb{B}.
\end{array}$$

$$\begin{array}{l}
\texttt{Inductive}\ \mathbb{B}_b := \\
\mid\ \mathit{true}_b : \mathbb{B}_b \\
\mid\ \mathit{false}_b : \mathbb{B}_b \\
\mid\ \beta_{\mathbb{B}_b} : (\mathbb{N} \to \mathbb{B}_b) \to \mathbb{N} \to \mathbb{B}_b.
\end{array}$$

$$\begin{array}{l}
\texttt{Inductive}\ \mathbb{B}_\epsilon^\alpha : \mathbb{B} \to \mathbb{B}_b \to \square_i := \\
\mid\ \mathit{true}_\epsilon^\alpha : \mathbb{B}_\epsilon^\alpha\ \mathit{true}\ \mathit{true}_b \\
\mid\ \mathit{false}_\epsilon^\alpha : \mathbb{B}_\epsilon^\alpha\ \mathit{false}\ \mathit{false}_b \\
\mid\ \beta_{\mathbb{B}_\epsilon^\alpha} : \forall\ (b_a : \mathbb{B}) \\
\qquad\qquad (f : \mathbb{N} \to \mathbb{B}_b) \\
\qquad\qquad (n : \mathbb{N}) \\
\qquad\qquad (b_\epsilon : \mathbb{B}_\epsilon^\alpha\ b_a\ (f\ (\alpha\ n))), \\
\qquad \mathbb{B}_\epsilon^\alpha\ b_a\ (\beta_{\mathbb{B}_b}\ f\ n).
\end{array}$$

Fundamental property : $\Pi(b_a : \mathbb{B})(b_b : \mathbb{B}_b)(b_\epsilon : \mathbb{B}_\epsilon^\alpha\ b_a\ b_b).\ b_a = \partial\ \alpha\ b_b$

V. The full model

Final theorem

Theorem

Given

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

in the source theory, then

$$\lambda\alpha.\ [f]_a^\alpha\ \alpha$$

is continuous in the target theory.

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]^{\alpha}_{a}\ \alpha : \mathbb{N}$ $\qquad\qquad$ $\vdash [f]_{b}\ \gamma_{b} : \mathbb{N}_{b}$

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_a^\alpha\ \alpha : \mathbb{N} \qquad\qquad \vdash [f]_b\ \gamma_b : \mathbb{N}_b$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_\epsilon^\alpha\ \gamma_\epsilon : \mathbb{N}_\epsilon\ ([f]_a^\alpha\ \alpha)([f]_b\ \gamma_b)$$

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_a^\alpha\ \alpha : \mathbb{N} \qquad \vdash [f]_b\ \gamma_b : \mathbb{N}_b$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_\epsilon^\alpha\ \gamma_\epsilon : \mathbb{N}_\epsilon\ ([f]_a^\alpha\ \alpha)([f]_b\ \gamma_b)$$

$$[f]_a^\alpha\ \alpha = \partial\ \alpha\ ([f]_b\ \gamma_b)$$

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_a^{\alpha}\ \alpha : \mathbb{N} \qquad\qquad \vdash [f]_b\ \gamma_b : \mathbb{N}_b$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_\epsilon^{\alpha}\ \gamma_\epsilon : \mathbb{N}_\epsilon\ ([f]_a^{\alpha}\ \alpha)([f]_b\ \gamma_b)$$

$$f\ \alpha = \partial\ \alpha\ ([f]_b\ \gamma_b)$$

V. The full model

A sense of déjà-vu (this subtitle is part of it)

$$\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_a^{\alpha}\ \alpha : \mathbb{N} \qquad\qquad \vdash [f]_b\ \gamma_b : \mathbb{N}_b$$

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash [f]_{\epsilon}^{\alpha}\ \gamma_{\epsilon} : \mathbb{N}_{\epsilon}\ ([f]_a^{\alpha}\ \alpha)([f]_b\ \gamma_b)$$

$$\Pi(\alpha : \mathbb{N} \to \mathbb{N}).\ f\ \alpha = \partial\ \alpha\ ([f]_b\ \gamma_b)$$

Future work

Future work

Going internal ?

Future work

Going internal ?

$$\begin{tikzcd}
\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \arrow[d, bend right, "\mathit{Meta}"'] \\
\vdash_{\mathsf{CIC}} \texttt{continuous}\ (\lambda \alpha.\ [f]_a^{\alpha}\ \alpha)
\end{tikzcd}$$

Future work

Going internal ?

$$\begin{tikzcd}
\vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \arrow[d, bend right, "\textit{Meta}"'] \\
\vdash_{\mathsf{CIC}} \mathtt{continuous}\ (\lambda\alpha.\ [f]^{\alpha}_{a}\ \alpha)
\end{tikzcd}$$

$$\begin{tikzcd}
\vdash_{\mathsf{BTT}} \mathsf{Axiom}\ \Phi : \Pi f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \mathtt{continuous}\ f \arrow[d, bend right, "\textit{Meta}"'] \\
\vdash_{\mathsf{CIC}} [\Phi]_{g} : [\![\Pi f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \mathtt{continuous}\ f]\!]_{g}
\end{tikzcd}$$

Future work

Going internal ?

$$\begin{tikzcd} \vdash_{\mathsf{BTT}} f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \arrow[d, bend right, "\mathit{Meta}"'] \\ \vdash_{\mathsf{CIC}} \mathtt{continuous}\ (\lambda\alpha.\ [f]_a^\alpha\ \alpha) \end{tikzcd}$$

$$\begin{tikzcd} \vdash_{\mathsf{BTT}} \mathsf{Axiom}\ \Phi : \Pi f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \mathtt{continuous}\ f \arrow[d, bend right, "\mathit{Meta}"'] \\ \vdash_{\mathsf{CIC}} [\Phi]_g : [\![\Pi f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \mathtt{continuous}\ f]\!]_g \end{tikzcd}$$

No-go theorem for CIC:

$$\vdash_{\mathsf{CIC}} (\Pi f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}.\ \mathtt{continuous}\ f) \to 0 = 1$$

Future work

Going CIC ?

BTT = Dependent Type Theory

with restricted dependent elimination to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, \mathbb{B}\_\mathrm{store}\ P\ b}$$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{true} \equiv P\ \mathtt{true}$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{false} \equiv P\ \mathtt{false}$

$\mathbb{B}\_\mathrm{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

Future work

Going CIC ?

BTT = Dependent Type Theory

with restricted dependent elimination to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathsf{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\ P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathsf{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\ \mathbb{B}\_\mathsf{store}\ P\ b}$$

$\mathbb{B}\_\mathsf{store}\ P\ \mathtt{true} \equiv P\ \mathtt{true}$

$\mathbb{B}\_\mathsf{store}\ P\ \mathtt{false} \equiv P\ \mathtt{false}$

$\mathbb{B}\_\mathsf{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

Ensure that P cannot discriminate between pure and effectful terms?

Future work

Going CIC ?

BTT = Dependent Type Theory
with restricted dependent elimination to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, \mathbb{B}\_\mathrm{store}\ P\ b}$$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{true} \equiv P\ \mathtt{true}$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{false} \equiv P\ \mathtt{false}$

$\mathbb{B}\_\mathrm{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

Ensure that P cannot discriminate between pure and effectful terms?

Quotients ?

Future work

Going CIC ?

BTT = Dependent Type Theory

with restricted dependent elimination to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \texttt{true} \qquad \vdash u_f : P\ \texttt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, \mathbb{B}\_\mathrm{store}\ P\ b}$$

$\mathbb{B}\_\mathrm{store}\ P\ \texttt{true} \equiv P\ \texttt{true}$

$\mathbb{B}\_\mathrm{store}\ P\ \texttt{false} \equiv P\ \texttt{false}$

$\mathbb{B}\_\mathrm{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

Ensure that P cannot discriminate between pure and effectful terms?

Quotients ?

Sheaves ?

Future work

Going CIC ?

BTT = Dependent Type Theory
with restricted dependent elimination to accommodate effects

CIC

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, P\ b}$$

BTT

$$\frac{\vdash P : \mathbb{B} \to \square \qquad \vdash u_t : P\ \mathtt{true} \qquad \vdash u_f : P\ \mathtt{false}}{\vdash \mathbb{B}\_\mathrm{rect}\ P\ u_t\ u_f : \Pi(b : \mathbb{B}).\, \mathbb{B}\_\mathrm{store}\ P\ b}$$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{true} \equiv P\ \mathtt{true}$

$\mathbb{B}\_\mathrm{store}\ P\ \mathtt{false} \equiv P\ \mathtt{false}$

$\mathbb{B}\_\mathrm{store}\ P\ \beta$ underspecified for any β non standard inhabitant of $\mathbb{B}$

Ensure that P cannot discriminate between pure and effectful terms?

Quotients ?

Sheaves ?

Use an other proof technique altogether?

Back to square 1

The axiomatic translation

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash \boldsymbol{t} : \mathbb{N}$$

Back to square 1

The axiomatic translation

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash \boldsymbol{t} : \mathbb{N}$$

Look at the structure of the term using NbE

Back to square 1

The axiomatic translation

$$\alpha : \mathbb{N} \to \mathbb{N} \vdash \boldsymbol{t} : \mathbb{N}$$

- Look at the structure of the term using NbE
- Get the branching structure from the term itself

Conclusion

Thank you for watching