The Fire Triangle

How to Mix Substitution, Dependent Elimination and Effects

Pierre-Marie Pédrot, Nicolas Tabareau

Gallinette (INRIA)

POPL'20

January, 23th 2020

It's Time to CIC Ass and Chew Bubble-Gum

CIC, the Calculus of Inductive Constructions.

It's Time to CIC Ass and Chew Bubble-Gum

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

The Pinnacle of the Curry-Howard correspondence

It's Time to CIC Ass and Chew Bubble-Gum

CIC, the Calculus of Inductive Constructions.

CIC, a very fancy **intuitionistic** **logical system**.

- Not just higher-order logic, not just first-order logic
- First class notion of computation and crazy inductive types

CIC, a very powerful **functional** **programming language**.

- Finest types to describe your programs
- No clear phase separation between runtime and compile time

The Pinnacle of the Curry-Howard correspondence

A CIC Joke

Yet CIC suffers from a **fundamental** flaw.

A CIC Joke

Yet CIC suffers from a fundamental flaw.

- You want to show the wonders of Coq to a fellow programmer
- You fire your favourite IDE
- … and you're asked the DREADFUL question.

A CIC Joke

Yet CIC suffers from a **fundamental** flaw.

- You want to show the wonders of Coq to a fellow programmer
- You fire your favourite IDE
- … and you're asked the DREADFUL question.

COULD YOU WRITE A HELLO WORLD?

Sad reality (a.k.a. Curry-Howard)

Intuitionistic Logic $\Leftrightarrow$ **Functional** Programming

Sad reality (a.k.a. Curry-Howard)

Intuitionistic Logic ⇔ Functional Programming

Coq is even purer than Haskell:

- No mutable state (obviously)
- No exceptions (Haskell has them somehow)
- No arbitrary recursion
- and also no **HELLO WORLD** !

Sad reality (a.k.a. Curry-Howard)

Intuitionistic Logic ⇔ **Functional** Programming

Coq is even purer than Haskell:

- No mutable state (obviously)
- No exceptions (Haskell has them somehow)
- No arbitrary recursion
- and also no HELLO WORLD !

We want a type theory with effects !

Not Not a Problem

Intuitionistic Logic $\Leftrightarrow$ **Functional** Programming

Not Not a Problem

Intuitionistic Logic $\Leftrightarrow$ **Functional** Programming

Thus, the same problem for mathematically inclined users.

Not Not a Problem

Intuitionistic Logic ⇔ **Functional** Programming

Thus, the same problem for mathematically inclined users.

HOW DO I REASON CLASSICALLY?

Not Not a Problem

Intuitionistic Logic ⇔ **Functional** Programming

Thus, the same problem for mathematically inclined users.

HOW DO I REASON CLASSICALLY?

Thesis

Non-Intuitionistic Logic $\Leftrightarrow$ **Impure** Programming

Thesis

Non-Intuitionistic Logic ⇔ **Impure** Programming

We want a type theory with effects!

Thesis

Non-Intuitionistic Logic ⇔ **Impure** Programming

We want a type theory with effects!

To program more!

- Non-termination
- Exceptions
- State...

To prove more!

- Classical logic
- Univalence
- Choice...

Something is Rotten in the State of Type Theory

Classical logic does not play well with type theory.

- Barthe and Uustalu: CPS cannot interpret dependent elimination
- **Herbelin's paradox: CIC + `callcc` is unsound!**

Something is Rotten in the State of Type Theory

Classical logic does not play well with type theory.

- Barthe and Uustalu: CPS cannot interpret dependent elimination
- **Herbelin's paradox: CIC + `callcc` is unsound!**

We have been working on effectful type theories

Our specialty:

Something is Rotten in the State of Type Theory

Classical logic does not play well with type theory.

- Barthe and Uustalu: CPS cannot interpret dependent elimination
- **Herbelin's paradox: CIC + `callcc` is unsound!**

We have been working on effectful type theories

Our specialty:

We justify them through program translations into CIC itself.

Forcing, reader monad, exceptions, free algebraic...

Something is Rotten in the State of Type Theory

Classical logic does not play well with type theory.

- Barthe and Uustalu: CPS cannot interpret dependent elimination
- **Herbelin's paradox: CIC + `callcc` is unsound!**

We have been working on effectful type theories

Our specialty:

We justify them through program translations into CIC itself.

Forcing, reader monad, exceptions, free algebraic...

Effectful theories are always half-broken

- dependent elimination has to be restricted (BTT)
- or consistency forsaken, or worse

I Have a Bad Feeling about This

Why do we have trouble mixing effects and dependent types?

I Have a Bad Feeling about This

Why do we have trouble mixing effects and dependent types?

Coincidence? I Think Not!

Definition

A type theory enjoys *substitution* if the following rule is derivable.

$$\frac{\Gamma, x : X \vdash \bullet : A \qquad \Gamma \vdash t : X}{\Gamma \vdash \bullet : A\{x := t\}}$$

Definition

A type theory enjoys *substitution* if the following rule is derivable.

$$\frac{\Gamma, x : X \vdash \bullet : A \qquad \Gamma \vdash t : X}{\Gamma \vdash \bullet : A\{x := t\}}$$

Definition

A type theory enjoys *dependent elimination* on booleans if we have:

$$\frac{\Gamma, b : \mathbb{B} \vdash P : \square \qquad \Gamma \vdash \bullet : P\{b := \texttt{true}\} \qquad \Gamma \vdash \bullet : P\{b := \texttt{false}\}}{\Gamma, b : \mathbb{B} \vdash \bullet : P}$$

Definition

A type theory enjoys *substitution* if the following rule is derivable.

$$\frac{\Gamma, x : X \vdash \bullet : A \qquad \Gamma \vdash t : X}{\Gamma \vdash \bullet : A\{x := t\}}$$

Definition

A type theory enjoys *dependent elimination* on booleans if we have:

$$\frac{\Gamma, b : \mathbb{B} \vdash P : \Box \qquad \Gamma \vdash \bullet : P\{b := \mathtt{true}\} \qquad \Gamma \vdash \bullet : P\{b := \mathtt{false}\}}{\Gamma, b : \mathbb{B} \vdash \bullet : P}$$

Definition

A type theory has *observable effects* if there is a closed term $t : \mathbb{B}$ that is **not observationally equivalent to a value**, i.e. there is a context $C[\cdot]$ s.t.

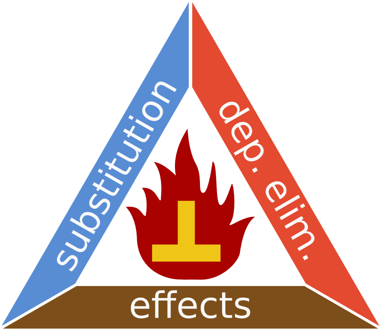
$$C[\mathtt{true}] \equiv \mathtt{true} \quad \text{and} \quad C[\mathtt{false}] \equiv \mathtt{true} \quad \text{but} \quad C[t] \equiv \mathtt{false}$$

Type Theory on Fire

Sounds like desirable properties, right?

Type Theory on Fire

Sounds like desirable properties, right?



Theorem (Fire Triangle)

substitution + dep. elimination + effects $\vdash$ *logically inconsistent.*

There Is No Free Lunch

The proof is actually straightforward.

Proof.

If C distinguishes boolean values from an effectful term M, prove by dependent elimination $\Pi(b : \mathbb{B}).\ C[b] = \texttt{false}$, apply to M and derive $\texttt{true} = \texttt{false}$. □

There Is No Free Lunch

The proof is actually straightforward.

Proof.

If C distinguishes boolean values from an effectful term M, prove by dependent elimination $\Pi(b : \mathbb{B}).\ C[b] = \texttt{false}$, apply to M and derive $\texttt{true} = \texttt{false}$. □

We essentially retrofitted the definition of effects to make it work.

There Is No Free Lunch

The proof is actually straightforward.

Proof.

If C distinguishes boolean values from an effectful term M, prove by dependent elimination $\Pi(b : \mathbb{B}).\ C[b] = \texttt{false}$, apply to M and derive $\texttt{true} = \texttt{false}$. □

We essentially retrofitted the definition of effects to make it work.

‘But most effects are also observables effects!

So it’s not cheating either.

There Is No Free Lunch

The proof is actually straightforward.

Proof.

If C distinguishes boolean values from an effectful term M, prove by dependent elimination $\Pi(b : \mathbb{B}).\ C[b] = \texttt{false}$, apply to M and derive $\texttt{true} = \texttt{false}$. $\square$

We essentially retrofitted the definition of effects to make it work.

‘But most effects are also observables effects!

So it’s not cheating either.

And now for a high-level overview of the problem and solutions

It is not a Bug, it is a Feature™

Dependent types entail one major difference with simpler type systems.

It is not a Bug, it is a Feature™

Dependent types entail one major difference with simpler type systems.

$$\frac{A \equiv_\beta B \qquad \Gamma \vdash M : B}{\Gamma \vdash M : A}$$

It is not a Bug, it is a Feature™

Dependent types entail one major difference with simpler type systems.

$$\frac{A \equiv_\beta B \qquad \Gamma \vdash M : B}{\Gamma \vdash M : A}$$

Bad news 1

Typing rules embed the dynamics of programs!

It is not a Bug, it is a Feature™

Dependent types entail one major difference with simpler type systems.

$$\frac{A \equiv_\beta B \qquad \Gamma \vdash M : B}{\Gamma \vdash M : A}$$

Bad news 1

Typing rules embed the dynamics of programs!

Bad news 2

Effects make reduction strategies relevant.

Call-by-name vs. Call-by-value

Call-by-name vs. Call-by-value

- Call-by-name: **functions** well-behaved vs. **inductives** ill-behaved
- Call-by-value: **inductives** well-behaved vs. **functions** ill-behaved

Call-by-name vs. Call-by-value

- Call-by-name: **functions** well-behaved vs. **inductives** ill-behaved
- Call-by-value: **inductives** well-behaved vs. **functions** ill-behaved

Substitution is a feature of call-by-name

Dependent elimination is a feature of call-by-value

Impossible is not French

Three knobs ⇒ **Four** solutions

Impossible is not French

Three knobs ⇒ **Four** solutions

- **Down with effects**: CBN and CBV reconcile

This is good ol' CIC, Keep Calm and Carry on. (†)

Impossible is not French

Three knobs $\Rightarrow$ **Four** solutions

- **Down with effects**: CBN and CBV reconcile

This is good ol' CIC, KEEP CALM AND CARRY ON. (†)

- **Go CBN** and restrict dependent elimination: Baclofen Type Theory

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 : \texttt{if } M \texttt{ then } P_1 \texttt{ else } P_2$$

Impossible is not French

Three knobs $\Rightarrow$ **Four** solutions

- **Down with effects**: CBN and CBV reconcile

This is good ol' CIC, KEEP CALM AND CARRY ON. (†)

- **Go CBN** and restrict dependent elimination: Baclofen Type Theory

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 : \texttt{if } M \texttt{ then } P_1 \texttt{ else } P_2$$

- **CBV rules**, respect values, and dump substitution

The least conservative approach

Impossible is not French

Three knobs ⇒ **Four** solutions

- **Down with effects**: CBN and CBV reconcile

This is good ol' CIC, KEEP CALM AND CARRY ON. (†)

- **Go CBN** and restrict dependent elimination: Baclofen Type Theory

$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2 : \texttt{if } M \texttt{ then } P_1 \texttt{ else } P_2$

- **CBV rules**, respect values, and dump substitution

The least conservative approach

- Who cares about consistency? **I want all!**

A paradigm shift: from type theory to dependent languages, e.g. ExTT

Pick Your Side, Comrade

Assuming you want consistent dependent effects...

Call-by-name vs. Call-by-value

Assuming you want consistent dependent effects...

~~**Call-by-n[illegible]-by-value**~~

Call-by-name **and** Call-by-value

CBPV

Assuming you want consistent dependent effects...

~~**Call-by-n[illegible]-by-value**~~

Call-by-name **and** Call-by-value

∂CBPV

(We had to pick a fancy name, everything else already taken.)

∂CBPV

Justified by all of our syntactic models so far

And we have quite a few!

- Impure Forcing — Unnatural Presheaves
- Reader
- Exceptions — Free algebraic effects
- Self-algebraic monads
- ...

∂CBPV

Justified by all of our syntactic models so far

And we have quite a few!

- Impure Forcing — Unnatural Presheaves
- Reader
- Exceptions — Free algebraic effects
- Self-algebraic monads
- … $\leftarrow$ notice the lack of CPS here

∂CBPV

The main novelties: two for the price of one

- Not one, but **two** parallel hierarchies of universes: $\square_v$ vs. $\square_c$!
- Not one, but **two** let-bindings!

$$\frac{\Gamma \vdash t : F\ A \qquad \color{red}{\Gamma \vdash X : \square_c} \qquad \Gamma, x : A \vdash u : X}{\Gamma \vdash \texttt{let}\ x := t\ \texttt{in}\ u : X}$$

$$\frac{\Gamma \vdash t : F\ A \qquad \color{red}{\Gamma, x : A \vdash X : \square_c} \qquad \Gamma, x : A \vdash u : X}{\Gamma \vdash \texttt{dlet}\ x := t\ \texttt{in}\ u : \texttt{let}\ x := t\ \texttt{in}\ X}$$

∂CBPV

The main novelties: two for the price of one

- Not one, but **two** parallel hierarchies of universes: $\square_v$ vs. $\square_c$!
- Not one, but **two** let-bindings!

$$\frac{\Gamma \vdash t : F\ A \qquad \Gamma \vdash X : \square_c \qquad \Gamma, x : A \vdash u : X}{\Gamma \vdash \texttt{let}\ x := t\ \texttt{in}\ u : X}$$

$$\frac{\Gamma \vdash t : F\ A \qquad \Gamma, x : A \vdash X : \square_c \qquad \Gamma, x : A \vdash u : X}{\Gamma \vdash \texttt{dlet}\ x := t\ \texttt{in}\ u : \texttt{let}\ x := t\ \texttt{in}\ X}$$

See the paper for more details

Much More

This was a very high-level talk

Many things I did not discuss here!

- A good notion of purity: thunkability vs. linearity
- Complex ∂CBPV encodings
- Explicit model constructions
- A new look on presheaves

Conclusion

What we did

- Effects and dependent types: you can't have your cake and eat it.
  - $\rightsquigarrow$ Purity, CBN, CBV, Michael Bay?
- Even inconsistent theories have practical interest.
- ∂CBPV a unifying framework for dependent effects

Conclusion

What we did

- Effects and dependent types: you can't have your cake and eat it.
  $\rightsquigarrow$ Purity, CBN, CBV, Michael Bay?
- Even inconsistent theories have practical interest.
- ∂CBPV a unifying framework for dependent effects

What we should probably do

- Study more in details CBV type theories
- Try to give a model for classical logic, choice, what else?
- Implement ∂CBPV?