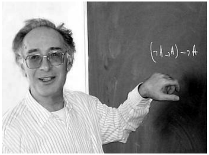
Dialectique concrète et machines abstraites

From Gödel...

Pierre-Marie Pédrot

PPS/πr^2

Journées PPS



... to Krivine

Once upon a time...

- Cataclysm: Gödel's incompleteness theorem (1931)

Once upon a time...

- Cataclysm: Gödel's incompleteness theorem (1931)

We do not fight alienation with an alienated logic.

Once upon a time...

- Cataclysm: Gödel's incompleteness theorem (1931)

We do not fight alienation with an alienated logic.

- Justifying arithmetic differently
- ... Intuitionistic logic!
  - The double-negation translation (1933)
  - The functional interpretation aka Dialectica (30's, published 1958)

What it is...

What is Dialectica?

What it is...

What is Dialectica?

- A translation $(-)^D$ from HA into HA^ω
- That preserves intuitionistic content

What it is...

What is Dialectica?

- A translation $(-)^D$ from HA into HA$^\omega$
- That preserves intuitionistic content
- But offers two additional semi-classical principles

$$\text{MP}\ \frac{\neg(\forall n \in \mathbb{N}.\, \neg P\, n)}{\exists n \in \mathbb{N}.\, P\, n} \qquad \frac{I \to \exists m \in \mathbb{N}.\, Q\, m}{\exists m \in \mathbb{N}.\, I \to Q\, m}\ \text{IP}$$

Markov's principle **Independence of premise**

(P decidable, I irrelevant)

... and what it is not.

What is not Dialectica?

... and what it is not.

What is not Dialectica?

Not a nice proof-theoretical translation...

- Only preserves provability, breaks β-equivalence!

$$t \equiv_\beta u \not\to t^D \equiv_\beta u^D$$

- Full of historical hacks from the dawn of proof theory
- Poorly understood as a program translation (side-effects)

In this talk

A modern, proof-theoretical,
Curry-Howardesque Dialectica.

In this talk

A modern, proof-theoretical,
Curry-Howardesque Dialectica.

- As a translation acting on the untyped λ-calculus
  - $\rightsquigarrow$ No arithmetical tricks!
- Calling-convention agnostic
  - $\rightsquigarrow$ Thanks to De Paiva's linear decomposition
- An operational explanation through the Krivine machine
  - $\rightsquigarrow$ Inspired by classical realizability & forcing à la Krivine
- Bonus: free extension to dependently typed systems

Historical presentation

$$\boxed{\vdash A \qquad \mapsto \qquad \vdash A^D \equiv \exists \vec{u}.\, \forall \vec{x}.\, A_D[\vec{u}, \vec{x}]}$$

- $(-)_D$ essentially commutes with the connectives

Historical presentation

$$\boxed{\vdash A \qquad \mapsto \qquad \vdash A^D \equiv \exists \vec{u}. \forall \vec{x}. A_D[\vec{u}, \vec{x}]}$$

- $(-)_D$ essentially commutes with the connectives
- ... except for the arrow! (stay tuned)

Theorem (Soundness)

If $\vdash_{HA} A$ *then* $\vdash_{HA^\omega} A^D$.

Dissecting the formula

$$\exists \vec{u}.\ \forall \vec{x}.\ A_D[\vec{u}, \vec{x}]$$

Witness	Counter	Orthogonality
$\mathbb{W}(A)$	$\mathbb{C}(A)$	$\perp_A$
`Type`	`Type`	$\mathbb{W}(A) \to \mathbb{C}(A) \to$ `Prop`

A proof $\vdash u : A$ is a term $\vdash u : \mathbb{W}(A)$ such that $\forall x : \mathbb{C}(A).\, u \perp_A x$

Linearized Dialectica

- We can even refine this picture
- We focus on propositional logic
- Dialectica factors through linear logic (De Paiva '89)

$$A \to B \quad := \quad !A \multimap B$$

- The historical version is call-by-name
  - $\leadsto$ ... but we can choose another decomposition
  - $\leadsto$ ... whose operational contents will make sense (later on)

The linear decomposition of the arrow

	$\mathbb{W}$	$\mathbb{C}$	$\bot$
$A \multimap B$	$\begin{cases} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{C}(A) \end{cases}$	$\mathbb{W}(A) \times \mathbb{C}(B)$	$\ldots$
$!A$	$\mathbb{W}(A)$	$\mathbb{W}(A) \to \mathbb{C}(A)$	$\ldots$
$A \to B$	$\begin{cases} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{W}(A) \to \mathbb{C}(A) \end{cases}$	$\mathbb{W}(A) \times \mathbb{C}(B)$	$\ldots$

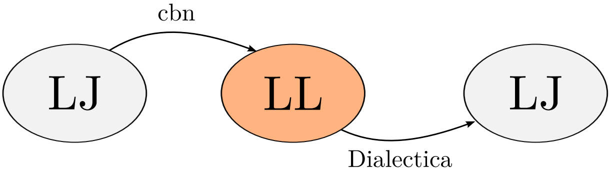
The linear decomposition of the arrow

	$\mathbb{W}$	$\mathbb{C}$	$\bot$
$A \multimap B$	$\begin{cases} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{C}(A) \end{cases}$	$\mathbb{W}(A) \times \mathbb{C}(B)$	$\ldots$
$!A$	$\mathbb{W}(A)$	$\mathbb{W}(A) \to \mathbb{C}(A)$	$\ldots$
$A \to B$	$\begin{cases} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{W}(A) \to \mathbb{C}(A) \end{cases}$	$\mathbb{W}(A) \times \mathbb{C}(B)$	$\ldots$

- Reversible arrows!
- Two arrows for the price of one!
- First-class stacks!

Intepretation of the call-by-name λ-calculus

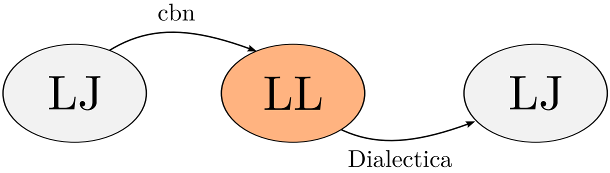
We are now trying to translate the λ-calculus through Dialectica.



- First through the call-by-name linear decomposition into LL
- Then into LJ with the linear Dialectica
- We are interested in the resulting composition

Intepretation of the call-by-name λ-calculus

We are now trying to translate the λ-calculus through Dialectica.



- First through the call-by-name linear decomposition into LL
- Then into LJ with the linear Dialectica
- We are interested in the resulting composition
- (No more LL in this talk, you can breathe easy)

Through the looking glass

We have the following nice isomorphism:

$$[\![x_1 : \Gamma_1, \ldots, x_n : \Gamma_n \vdash t : A]\!] \cong \mathbb{W}(\Gamma) \to \begin{cases} \mathbb{W}(A) \\ \mathbb{C}(A) \to \mathbb{C}(\Gamma_1) \\ \vdots \\ \mathbb{C}(A) \to \mathbb{C}(\Gamma_n) \end{cases}$$

Through the looking glass

We have the following nice isomorphism:

$$[\![x_1 : \Gamma_1, \ldots, x_n : \Gamma_n \vdash t : A]\!] \cong \mathbb{W}(\Gamma) \to \begin{cases} \mathbb{W}(A) \\ \mathbb{C}(A) \to \mathbb{C}(\Gamma_1) \\ \vdots \\ \mathbb{C}(A) \to \mathbb{C}(\Gamma_n) \end{cases}$$

Which results in the following translations:

$$[\![\vec{x} : \Gamma \vdash t : A]\!] \equiv \begin{cases} \vec{x} : \mathbb{W}(\Gamma) \vdash t^\bullet : \mathbb{W}(A) \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_1} : \mathbb{C}(A) \to \mathbb{C}(\Gamma_1) \\ \vdots \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_n} : \mathbb{C}(A) \to \mathbb{C}(\Gamma_n) \end{cases}$$

A glimpse at the translation

For $(-)^{\bullet}$: $\quad \Gamma \vdash t : A \quad \mapsto \quad \mathbb{W}(\Gamma) \vdash t^{\bullet} : \mathbb{W}(A)$

$$
\begin{array}{lcl}
x^{\bullet} & \equiv & x \\
(\lambda x.\, t)^{\bullet} & \equiv & \left\{ \begin{array}{l} \lambda x.\, t^{\bullet} \\ \lambda \pi x.\, t_x\, \pi \end{array} \right. \\
(t\, u)^{\bullet} & \equiv & (\mathtt{fst}\ t^{\bullet})\, u^{\bullet}
\end{array}
$$

Artifacts

In order to interpret the $(-)_x$ translation, we need the following:

Dummy term

For all type A, there exists $\vdash \varnothing_A : \mathbb{C}(A)$.

Decidability of the orthogonality

For all A there exist some λ-term

$$@^A : \mathbb{C}(A) \to \mathbb{C}(A) \to \mathbb{W}(A) \to \mathbb{C}(A)$$

with the following behaviour:

$$\pi_1 @_x^A \pi_2 \cong \texttt{if } x \perp_A \pi_1 \texttt{ then } \pi_2 \texttt{ else } \pi_1$$

Translation, next

For t_x : $\Gamma \vdash t : A \quad \mapsto \quad \mathbb{W}(\Gamma) \vdash t_{x_i} : \mathbb{C}(A) \rightarrow \mathbb{C}(\Gamma_i)$

$$
\begin{array}{lcl}
x_x & \equiv & \lambda \pi.\, \pi \\
 & : & \mathbb{C}(A) \rightarrow \mathbb{C}(A) \\
y_x & \equiv & \lambda \pi.\, \varnothing \\
 & : & \mathbb{C}(A) \rightarrow \mathbb{C}(\Gamma_i) \\
(\lambda y.\, t)_x & \equiv & \lambda (y, \pi).\, t_x\, \pi \\
 & : & \mathbb{W}(A) \times \mathbb{C}(B) \rightarrow \mathbb{C}(\Gamma_i)
\end{array}
$$

Translation, next

For t_x : $\Gamma \vdash t : A \quad \mapsto \quad \mathbb{W}(\Gamma) \vdash t_{x_i} : \mathbb{C}(A) \rightarrow \mathbb{C}(\Gamma_i)$

$$
\begin{array}{lcl}
x_x & \equiv & \lambda \pi.\, \pi \\
 & : & \mathbb{C}(A) \rightarrow \mathbb{C}(A) \\
y_x & \equiv & \lambda \pi.\, \varnothing \\
 & : & \mathbb{C}(A) \rightarrow \mathbb{C}(\Gamma_i) \\
(\lambda y.\, t)_x & \equiv & \lambda (y, \pi).\, t_x\, \pi \\
 & : & \mathbb{W}(A) \times \mathbb{C}(B) \rightarrow \mathbb{C}(\Gamma_i) \\
(t\, u)_x & \equiv & \lambda \pi.\, u_x\, ((\texttt{snd}\ t^\bullet)\ \pi\, u^\bullet)\ @_\pi\ t_x\, (u^\bullet, \pi) \\
 & : & \mathbb{C}(B) \rightarrow \mathbb{C}(\Gamma_i)
\end{array}
$$

It just works... Does it?

Soundness

If $\vdash t : A$, then:

- $\vdash t^\bullet : \mathbb{W}(A)$
- for all $\pi : \mathbb{C}(A)$, $t^\bullet \perp_A \pi$.

It just works... Does it?

Soundness

If $\vdash t : A$, then:

- $\vdash t^\bullet : \mathbb{W}(A)$
- for all $\pi : \mathbb{C}(A)$, $t^\bullet \perp_A \pi$.

Sadness

The translation is still not stable by β-reduction.

Almost there

Using $\varnothing$ and @ is an encoding of Dialectica.

Almost there

Using $\varnothing$ and @ is an encoding of Dialectica.

- We want multisets $\mathfrak{M}$ (think of lists)!
- We just change:

$$\begin{array}{rcl} \mathbb{C}(!A) & \equiv & \mathbb{W}(A) \rightarrow \mathbb{C}(A) \\ \mathbb{C}(!A) & \equiv & \mathbb{W}(A) \rightarrow \mathfrak{M}\ \mathbb{C}(A) \end{array}$$

- Term interpretation is almost unchanged:
  - $\varnothing$ becomes the empty set: $\varnothing : \mathbb{C}(A)$ $\varnothing : \mathfrak{M}\ \mathbb{C}(A)$
  - @ becomes union:

$$\begin{array}{rl} @: & \mathbb{C}(A) \rightarrow \mathbb{C}(A) \rightarrow \mathbb{W}(A) \rightarrow \mathbb{C}(A) \\ @: & \mathfrak{M}\ \mathbb{C}(A) \rightarrow \mathfrak{M}\ \mathbb{C}(A) \rightarrow \mathfrak{M}\ \mathbb{C}(A) \end{array}$$

  - ... plus a bit of monadic boilerplate
- We do not need orthogonality anymore...

What about the computational content?

This gives us the following types for the translation:

$$[\![\vec{x} : \Gamma \vdash t : A]\!] \equiv \begin{cases} \vec{x} : \mathbb{W}(\Gamma) \vdash t^{\bullet} : \mathbb{W}(A) \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_1} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_1) \\ \vdots \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_n} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_n) \end{cases}$$

What about the computational content?

This gives us the following types for the translation:

$$[\![\vec{x} : \Gamma \vdash t : A]\!] \equiv \begin{cases} \vec{x} : \mathbb{W}(\Gamma) \vdash t^\bullet : \mathbb{W}(A) \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_1} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_1) \\ \vdots \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_n} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_n) \end{cases}$$

- $t^\bullet$ is clearly the lifting of t;

What about the computational content?

This gives us the following types for the translation:

$$[\![\vec{x} : \Gamma \vdash t : A]\!] \equiv \begin{cases} \vec{x} : \mathbb{W}(\Gamma) \vdash t^\bullet : \mathbb{W}(A) \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_1} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_1) \\ \vdots \\ \vec{x} : \mathbb{W}(\Gamma) \vdash t_{x_n} : \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_n) \end{cases}$$

- $t^\bullet$ is clearly the lifting of t;
- What on earth is t_{x_i}?

An unbearable suspense

A small interlude to introduce you to the KAM.

An unbearable suspense

A small interlude to introduce you to the KAM.

Closures	c	$::=$	(t, σ)
Environments	σ	$::=$	$\emptyset \mid \sigma + (x := c)$
Stacks	π	$::=$	$\varepsilon \mid c \cdot \pi$
Processes	p	$::=$	$\langle (t, \sigma) \mid \pi \rangle$

Push	$\langle (t\, u, \sigma) \mid \pi \rangle$	$\rightarrow$	$\langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle$
Pop	$\langle (\lambda x.\, t, \sigma) \mid c \cdot \pi \rangle$	$\rightarrow$	$\langle (t, \sigma + (x := c)) \mid \pi \rangle$
Grab	$\langle (x, \sigma + (x := c)) \mid \pi \rangle$	$\rightarrow$	$\langle c \mid \pi \rangle$
Garbage	$\langle (x, \sigma + (y := c)) \mid \pi \rangle$	$\rightarrow$	$\langle (x, \sigma) \mid \pi \rangle$

The Krivine Machine™

Fiat lux

Let $\langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle$ be a process. We get:

$$\vec{x} : \mathbb{W}(\Gamma) \vdash s_{x_i} : \mathbb{C}(A) \to \mathfrak{M}\,\mathbb{C}(\Gamma_i) \qquad \vdash \vec{r}^{\bullet} : \mathbb{W}(\Gamma) \qquad \vdash \pi^{\bullet} : \mathbb{C}(A)$$

Fiat lux

Let $\langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle$ be a process. We get:

$$\vec{x} : \mathbb{W}(\Gamma) \vdash s_{x_i} : \mathbb{C}(A) \to \mathfrak{M}\, \mathbb{C}(\Gamma_i) \qquad \vdash \vec{r}^{\bullet} : \mathbb{W}(\Gamma) \qquad \vdash \pi^{\bullet} : \mathbb{C}(A)$$

Then $s_{x_i}\{\vec{x} := \vec{r}^{\bullet}\}\, \pi^{\bullet}$ is the multiset made of the stacks encountered by x_i while evaluating $\langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle$, i.e.

$$(s_{x_i}\{\vec{x} := \vec{r}^{\bullet}\})\, \pi^{\bullet} = [\rho_1^{\bullet}; \ldots; \rho_m^{\bullet}]$$

$$\begin{array}{ccc} \langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle & \longrightarrow^{*} & \langle (x_i, \sigma_1) \mid \rho_1 \rangle \\ & \vdots & \vdots \\ & \longrightarrow^{*} & \langle (x_i, \sigma_m) \mid \rho_m \rangle \end{array}$$

Fiat lux

Let $\langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle$ be a process. We get:

$$\vec{x} : \mathbb{W}(\Gamma) \vdash s_{x_i} : \mathbb{C}(A) \to \mathfrak{M}\, \mathbb{C}(\Gamma_i) \qquad \vdash \vec{r}^\bullet : \mathbb{W}(\Gamma) \qquad \vdash \pi^\bullet : \mathbb{C}(A)$$

Then $s_{x_i}\{\vec{x} := \vec{r}^\bullet\}\, \pi^\bullet$ is the multiset made of the stacks encountered by x_i while evaluating $\langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle$, i.e.

$$(s_{x_i}\{\vec{x} := \vec{r}^\bullet\})\, \pi^\bullet = [\rho_1^\bullet; \ldots ; \rho_m^\bullet]$$

$$\begin{array}{rcl} \langle (s, (\vec{x} := \vec{r})) \mid \pi \rangle & \longrightarrow^* & \langle (x_i, \sigma_1) \mid \rho_1 \rangle \\ & \vdots & \vdots \\ & \longrightarrow^* & \langle (x_i, \sigma_m) \mid \rho_m \rangle \end{array}$$

Dialectica tracks accesses to the variables (Grab rule).

An application

$$(t\,u)_x \equiv \lambda\pi.\,(((\texttt{snd}\ t^\bullet)\ \pi\ u^\bullet) \gg\!= u_x)\ @\ t_x\ (u^\bullet, \pi)$$
$$: \mathbb{C}(B) \to \mathfrak{M}\ \mathbb{C}(\Gamma_i)$$

$$
\begin{array}{lll|l}
\langle (t\,u, \sigma) \mid \pi \rangle & \to & \langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle & t_x\ (u^\bullet, \pi) \\
 & \to^* & \langle (\lambda y.\hat{t}, \hat{\sigma}) \mid (u, \sigma) \cdot \pi \rangle & \\
 & \to & \langle (\hat{t}, \hat{\sigma} + (y := (u, \sigma))) \mid \pi \rangle & (\texttt{snd}\ t^\bullet)\ \pi\ u^\bullet \\
 & \to^* & \langle (y, \hat{\sigma} + (y := (u, \sigma)) \mid \rho_1 \rangle & u_x\ \rho_1 \\
 & & \ldots & \\
 & \to^* & \langle (y, \hat{\sigma} + (y := (u, \sigma)) \mid \rho_n \rangle & u_x\ \rho_n \\
 & & \ldots &
\end{array}
$$

Dialectica Reloaded

- The standard Dialectica only returns one stack
  - $\rightsquigarrow$ the first correct stack, dynamically tested

Dialectica Reloaded

- The standard Dialectica only returns one stack
  - $\rightsquigarrow$ the first correct stack, dynamically tested
- This is somehow a weak form of delimited control
  - $\rightsquigarrow$ Inspectable stacks: $\sim A$ $(:= \mathbb{C}(A))$ vs. $\neg A$ $(:= \mathbb{W}(\neg A))$
  - $\rightsquigarrow$ First class access to those stacks with $(-)_x$

Dialectica Reloaded

- The standard Dialectica only returns one stack
  - $\rightsquigarrow$ the first correct stack, dynamically tested
- This is somehow a weak form of delimited control
  - $\rightsquigarrow$ Inspectable stacks: $\sim A$ $(:= \mathbb{C}(A))$ vs. $\neg A$ $(:= \mathbb{W}(\neg A))$
  - $\rightsquigarrow$ First class access to those stacks with $(-)_x$
- We can do the same thing with other calling conventions

Something fishy

Actually, there is a subtle issue.

Something fishy

Actually, there is a subtle issue.

Something fishy

Actually, there is a subtle issue.

- Produced stacks are the right ones...

Something fishy

Actually, there is a subtle issue.

- Produced stacks are the right ones...
- They have the right multiplicity...

Something fishy

Actually, there is a subtle issue.

- Produced stacks are the right ones...
- They have the right multiplicity...
- But we lost the sequential order of the KAM!
- Because we used multisets (vs. lists)!

Something fishy

Actually, there is a subtle issue.

- Produced stacks are the right ones...
- They have the right multiplicity...
- But we lost the sequential order of the KAM!
- Because we used multisets (vs. lists)!
- Alas, no way to solve it without changing totally Dialectica.

Something fishy

Actually, there is a subtle issue.

- Produced stacks are the right ones...
- They have the right multiplicity...
- But we lost the sequential order of the KAM!
- Because we used multisets (vs. lists)!
- Alas, no way to solve it without changing totally Dialectica.

The faulty one is the application case (more generally duplication).

$$(t\,u)_x \equiv \lambda\pi.\,(((\texttt{snd}\ t^\bullet)\ \pi\ u^\bullet) \gg= u_x)\ @\ t_x\ (u^\bullet, \pi)$$

Towards CC$^\omega$

- What about more expressive systems?
- We follow the computation intuition we presented
- ... and we apply Dialectica to dependent types
  - $\rightsquigarrow$ subsuming first-order logic;
  - $\rightsquigarrow$ a proof-relevant $\forall$;
  - $\rightsquigarrow$ towards CC^ω and further!

Main lines

- We keep the CBN λ-calculus
  - $\rightsquigarrow$ it can be lifted readily to dependent types
  - $\rightsquigarrow$ $A \to B$ becomes $\Pi x : A.\, B$
  - $\rightsquigarrow$ $A \times B$ becomes $\Sigma x : A.\, B$
  - $\rightsquigarrow$ nothing special to do!

Main lines

- We keep the CBN λ-calculus
  - $\leadsto$ it can be lifted readily to dependent types
  - $\leadsto$ $A \to B$ becomes $\Pi x : A. B$
  - $\leadsto$ $A \times B$ becomes $\Sigma x : A. B$
  - $\leadsto$ nothing special to do!
- Design choice: a type A has no computational content:

$$A^\bullet \equiv (\mathbb{W}(A), \mathbb{C}(A)) : \texttt{Type} \times \texttt{Type}$$

$$A_x \equiv \lambda\pi.\emptyset \qquad \text{(effect-free)}$$

Main lines

- We keep the CBN λ-calculus
  - $\rightsquigarrow$ it can be lifted readily to dependent types
  - $\rightsquigarrow$ $A \rightarrow B$ becomes $\Pi x : A.\, B$
  - $\rightsquigarrow$ $A \times B$ becomes $\Sigma x : A.\, B$
  - $\rightsquigarrow$ nothing special to do!
- Design choice: a type A has no computational content:

$$A^\bullet \equiv (\mathbb{W}(A), \mathbb{C}(A)) : \texttt{Type} \times \texttt{Type}$$

$$A_x \equiv \lambda\pi.\emptyset \qquad \text{(effect-free)}$$

  - $\rightsquigarrow$ a bit disappointing;
  - $\rightsquigarrow$ but it works...
  - $\rightsquigarrow$ and the usual CC presentation does not help much!

Conclusion

- Actually, Dialectica is quite simple.
  - $\rightsquigarrow$ ... at least once we removed encoding artifacts

Conclusion

- Actually, Dialectica is quite simple.
  - ⇝ ... at least once we removed encoding artifacts
- It is a weak form of delimited control (the $(-)_x$ part)
  - ⇝ First-class inspectable stacks!
  - ⇝ Can be seen as a control operator

$$\mathscr{D} : (A \to B) \to A \to \sim B \to \mathfrak{M}(\sim A)$$

- But is is partially wrong:
  - ⇝ It is oblivious of sequentiality. How can we fix it?
  - ⇝ Related to the over-commutativity of LL

Conclusion

- Actually, Dialectica is quite simple.
  - $\rightsquigarrow$ ... at least once we removed encoding artifacts
- It is a weak form of delimited control (the $(-)_x$ part)
  - $\rightsquigarrow$ First-class inspectable stacks!
  - $\rightsquigarrow$ Can be seen as a control operator

$$\mathscr{D} : (A \to B) \to A \to {\sim}B \to \mathfrak{M}({\sim}A)$$

- But is is partially wrong:
  - $\rightsquigarrow$ It is oblivious of sequentiality. How can we fix it?
  - $\rightsquigarrow$ Related to the over-commutativity of LL
- The delimited control part can be lifted seamlessly to CC^ω
- Hindsight into categorical constructions? Relation to forcing?

Conclusion

- Actually, Dialectica is quite simple.
  - ⇝ ... at least once we removed encoding artifacts
- It is a weak form of delimited control (the $(-)_x$ part)
  - ⇝ First-class inspectable stacks!
  - ⇝ Can be seen as a control operator

$$\mathscr{D} : (A \to B) \to A \to \sim B \to \mathfrak{M}(\sim A)$$

- But is is partially wrong:
  - ⇝ It is oblivious of sequentiality. How can we fix it?
  - ⇝ Related to the over-commutativity of LL
- The delimited control part can be lifted seamlessly to CC^{ω}
- Hindsight into categorical constructions? Relation to forcing?

The hereabove illustrating assertions are non contractual.

Scribitur ad narrandum, non ad probandum

Thanks for your attention.