From Gödel to Curry-Howard

Pierre-Marie Pédrot

PPS/πr^2

TYPES 2014

Once upon a time...

- Cataclysm: Gödel's incompleteness theorem (1931)

- Cataclysm: Gödel's incompleteness theorem (1931)

$\boxed{\text{We do not fight alienation with an alienated logic.}}$

Once upon a time...

- Cataclysm: Gödel's incompleteness theorem (1931)

$\boxed{\text{We do not fight alienation with an alienated logic.}}$

- Justifying arithmetic differently
- ... Intuitionistic logic!
  - Double-negation translation (1933)
  - **Dialectica** (30's, published in 1958)

What is Dialectica?

What is Dialectica?

- A translation from HA into HA^ω
- That preserves intuitionistic content

What is Dialectica?

- A translation from HA into HA^ω
- That preserves intuitionistic content
- But offers two semi-classical principles:

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

Parental advisory required

For the sake of exhaustivity, we'll take a glimpse at the historical presentation of Dialectica.

Parental advisory required

For the sake of exhaustivity, we'll take a glimpse at the historical presentation of Dialectica.

Warning! Dusty logic inside

- Translation acting on formulæ
- Prevalence of negative connectives
- First-order logic
- Lots of arithmetic encoding
- Does not preserve β-reduction

Dialectica, Dawn of Curry-Howard:

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

$$\begin{array}{c|lll}
A \wedge B & \exists \vec{u}\, \vec{v}. & \forall \vec{x}\, \vec{y}. & A_D[\vec{u}, \vec{x}] \wedge B_D[\vec{v}, \vec{y}] \\
A \vee B & \exists \vec{u}\, \vec{v}\, b. & \forall \vec{x}\, \vec{y}. & (b = 0 \wedge A_D[\vec{u}, \vec{x}]) \vee (b = 1 \wedge B_D[\vec{v}, \vec{y}]) \\
A \to B & \exists \vec{\varphi}\, \vec{\psi}. & \forall \vec{u}\, \vec{y}. & A_D[\vec{u}, \vec{\psi}(\vec{u}, \vec{y})] \to B_D[\vec{\varphi}(\vec{u}), \vec{y}] \\
 & & & \\
\forall n.\, A[n] & \exists \vec{\varphi}. & \forall \vec{x}\, n. & A_D[\vec{\varphi}(n), \vec{x}, n] \\
\exists n.\, A[n] & \exists \vec{u}\, n. & \forall \vec{x}. & A_D[\vec{u}, n, \vec{x}]
\end{array}$$

Sound translation, blah blah blah.

A step into modernity

Let us forget the 50's, and rather jump directly to the 90's.

- Take seriously the **computational** content
- Dialectica as a **typed** object
- Works of De Paiva, Hyland, etc.

Types, types, types!

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

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

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

- We could give a computational content right now

Linearized Dialectica

- We could give a computational content right now
- But it would be *ad-hoc*, inheriting from the encodings of Dialectica
- Let us use our our favorite tool: **Linear Logic**!

- Call-by-name decomposition of the arrow:

$$A \to B \quad \equiv \quad !A \multimap B$$

Linearized Dialectica

- We could give a computational content right now
- But it would be *ad-hoc*, inheriting from the encodings of Dialectica
- Let us use our our favorite tool: **Linear Logic**!
- Call-by-name decomposition of the arrow:

$$A \to B \quad \equiv \quad !A \multimap B$$

Now we will be translating LL formulæ into LJ ones.

Requirements

- We will be interpreting the formulæ of linear logic:

$$A, B ::= A \otimes B \mid A \parr B \mid !A \mid ?A \mid A \oplus B \mid A \,\&\, B$$

- Sufficient to define $\mathbb{W}(A)$, $\mathbb{C}(A)$ and $\perp_A$
- Duality for free:
  - $\mathbb{W}(A^\perp) \equiv \mathbb{C}(A)$ and conversely
  - Orthogonal by complementation:

$$\frac{u \not\perp_A x}{x \perp_{A^\perp} u}$$

Linear decomposition

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

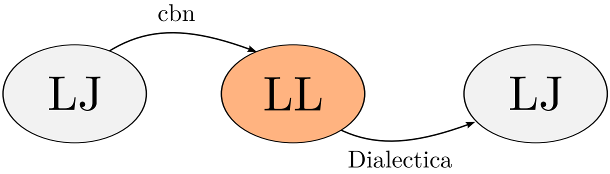
Linear decomposition

	$\mathbb{W}$	$\mathbb{C}$
$A \to B$	$\left\{ \begin{array}{c} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{W}(A) \to \mathbb{C}(A) \end{array} \right.$	$\mathbb{W}(A) \times \mathbb{C}(B)$
$A \multimap B$	$\left\{ \begin{array}{c} \mathbb{W}(A) \to \mathbb{W}(B) \\ \mathbb{C}(B) \to \mathbb{C}(A) \end{array} \right.$	$\mathbb{W}(A) \times \mathbb{C}(B)$
$!A$	$\mathbb{W}(A)$	$\mathbb{W}(A) \to \mathbb{C}(A)$

$$\frac{u \perp_A \psi\, y \quad \to \quad \varphi\, u \perp_B y}{(\varphi, \psi) \perp_{A \multimap B} (u, y)} \qquad \frac{u \perp_A z\, u}{u \perp_{!A} z}$$

Intepretation of the call-by-name λ-calculus

We're 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.

Brief reminder

We recall here the call-by-name translation of the λ-calculus into LL:

$$[\![A \to B]\!] \equiv ![\![A]\!] \multimap [\![B]\!]$$

$$[\![A \times B]\!] \equiv ![\![A]\!] \otimes ![\![B]\!]$$

$$[\![A + B]\!] \equiv ![\![A]\!] \oplus ![\![B]\!]$$

$$[\![\Gamma \vdash A]\!] \equiv \bigotimes ![\![\Gamma]\!] \vdash [\![A]\!]$$

Prolegomena

In order to interpret the λ-calculus, we need the following:

Dummy term

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

Decidability of the orthogonality

The $\perp_A$ relation is decidable. In particular, there must exist some λ-term

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

with the following behaviour:

$$u_1 @_x^A u_2 \cong \texttt{if } u_1 \perp_A x \texttt{ then } u_2 \texttt{ else } u_1$$

Did you solve the organization issue?

If we were to use the translation as is, we would bump up into an unbearable bureaucracy. Instead, we are going to use the following isomorphism.

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

Did you solve the organization issue?

If we were to use the translation as is, we would bump up into an unbearable bureaucracy. Instead, we are going to use the following isomorphism.

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

Which results in the following translations:

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

Translation

For $(-)^\bullet$:

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

Translation II

For t_x :

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

It just works... Does it?

Soundness

If $\vdash t : A$, then $\vdash t^\bullet : \mathbb{W}(A)$, and in addition, for all $\pi : \mathbb{C}(A)$, $t^\bullet \perp_A \pi$.

It just works... Does it?

Soundness

If $\vdash t : A$, then $\vdash t^\bullet : \mathbb{W}(A)$, and in addition, for all $\pi : \mathbb{C}(A)$, $t^\bullet \perp_A \pi$.

Sadness

The translation is still not stable by β-reduction.

Almost there

$\boxed{\text{Using } \varnothing \text{ and } @ \text{ is another encoding of Dialectica.}}$

Almost there

$\boxed{\text{Using } \varnothing \text{ and } @ \text{ is another encoding of Dialectica.}}$

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

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

- Term interpretation is almost unchanged:
  - $\varnothing$ becomes the empty set;
  - $@$ becomes union
  - ... 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 of ~~advertisement~~ **definitions** to introduce you to the KAM.

A small interlude of ~~advertisement~~ **definitions** 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	$::=$	$\langle c \mid \pi \rangle$

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™

Closures all the way down

Let:

- a term $\vec{x} : \Gamma \vdash t : A$
- a closure $\sigma \vdash \Gamma$
- a stack $\vdash \pi : A^{\perp}$ (i.e. $\pi^{\bullet} : \mathbb{C}(A)$)

Closures all the way down

Let:

- a term $\vec{x} : \Gamma \vdash t : A$
- a closure $\sigma \vdash \Gamma$
- a stack $\vdash \pi : A^{\perp}$ (i.e. $\pi^{\bullet} : \mathbb{C}(A)$)

Then $t_{x_i}\,\pi^{\bullet}$ is the multiset made of **the stacks encountered by** x_i while evaluating $\langle (t, \sigma) \mid \pi \rangle$, i.e.

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

$$\begin{array}{ccc} \langle (t, \sigma) \mid \pi \rangle & \longrightarrow^* & \langle (x_i, \sigma_1) \mid \rho_1 \rangle \\ & \vdots & \vdots \\ & \longrightarrow^* & \langle (x_i, \sigma_m) \mid \rho_m \rangle \end{array}$$

Otherwise said, Dialectica tracks the Grab rule.

Look

$$
\begin{array}{lcl}
x_x & \equiv & \lambda\pi.\,[\pi] \\
 & : & \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(A) \\
y_x & \equiv & \lambda\pi.\,[\,] \\
 & : & \mathbb{C}(A) \to \mathfrak{M}\ \mathbb{C}(\Gamma_i) \\
(\lambda y.\,t)_x & \equiv & \lambda(y,\pi).\,t_x\,\pi \\
 & : & \mathbb{W}(A) \times \mathbb{C}(B) \to \mathfrak{M}\ \mathbb{C}(\Gamma_i) \\
(t\,u)_x & \equiv & \lambda\pi.\,(((\mathtt{snd}\ t^\bullet)\,\pi\,u^\bullet) \gg\!= u_x)\ @\ t_x\,(u^\bullet,\pi) \\
 & : & \mathbb{C}(B) \to \mathfrak{M}\ \mathbb{C}(\Gamma_i)
\end{array}
$$

(We can generalize this interpretation to algebraic datatypes.)

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: $\sim A$ vs. $\neg A$
  - $\rightsquigarrow$ First class access to those stacks with $(-)_x$
  - $\rightsquigarrow$ Or through a control operator

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

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: $\sim A$ vs. $\neg A$
  - $\rightsquigarrow$ First class access to those stacks with $(-)_x$
  - $\rightsquigarrow$ Or through a control operator

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

- We can do the same thing with other calling conventions
  - $\rightsquigarrow$ The protohistoric Dialectica was call-by-name
  - $\rightsquigarrow$ Choose your favorite translation into LL!

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).

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

- 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
  - $\rightsquigarrow$ 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!
- Design choice: types have no computational content (effect-free):
  - $\rightsquigarrow$ a bit disappointing;
  - $\rightsquigarrow$ but it works...
  - $\rightsquigarrow$ and the usual CC presentation does not help much!

Type translation

Idea: if A is a type,

$$\begin{array}{lcl} A^\bullet \equiv & (\mathbb{W}(A), \mathbb{C}(A)) : \mathtt{Type} \times \mathtt{Type} & \\ A_x \equiv & \lambda\pi.\, [] & \text{(effect-free)} \end{array}$$

Type translation

Idea: if A is a type,

$$\begin{array}{rcll} A^\bullet & \equiv & (\mathbb{W}(A), \mathbb{C}(A)) : \mathtt{Type} \times \mathtt{Type} & \\ A_x & \equiv & \lambda\pi.\,[] & \text{(effect-free)} \end{array}$$

We get:

$$\begin{array}{lcl} \mathtt{Type}^\bullet & \equiv & (\mathtt{Type} \times \mathtt{Type}, 1) \\ \mathtt{Type}_x & \equiv & \lambda\pi.\,[\,] \\ (\Pi y : A.\, B)^\bullet & \equiv & \left(\begin{array}{c} (\Pi y : \mathbb{W}(A).\, \mathbb{W}(B)) \\ \times \\ (\Pi y : \mathbb{W}(A).\, \mathbb{C}(B) \to \mathfrak{M}\ \mathbb{C}(A)) \end{array}, \Sigma y : \mathbb{W}(A).\, \mathbb{C}(B) \right) \\ (\Pi y : A.\, B)_x & \equiv & \lambda\pi.\,[\,] \end{array}$$

(We can obtain inductives + dependent destruction quite easily.)

Conclusion

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

Conclusion

- Actually, Dialectica is quite simple.
  - $\rightsquigarrow$ ... at least once we removed encoding artifacts
- It is an approximation of ~~one~~ two side-effects:
  - $\rightsquigarrow$ A bit of delimited control (the $(-)_x$ part)
  - $\rightsquigarrow$ (A form of exceptions (with $\varnothing$))

Conclusion

- Actually, Dialectica is quite simple.
  - $\rightsquigarrow$ ... at least once we removed encoding artifacts
- It is an approximation of ~~one~~ two side-effects:
  - $\rightsquigarrow$ A bit of delimited control (the $(-)_x$ part)
  - $\rightsquigarrow$ (A form of exceptions (with $\varnothing$))
- But is is partially wrong:
  - $\rightsquigarrow$ it is oblivious of sequentiality
  - $\rightsquigarrow$ how can we fix it?

Conclusion

- Actually, Dialectica is quite simple.
  - $\leadsto$ … at least once we removed encoding artifacts
- It is an approximation of ~~one~~ two side-effects:
  - $\leadsto$ A bit of delimited control (the $(-)_x$ part)
  - $\leadsto$ (A form of exceptions (with $\varnothing$))
- But is is partially wrong:
  - $\leadsto$ it is oblivious of sequentiality
  - $\leadsto$ how can we fix it?
- The delimited control part can be lifted seamlessly to CC^ω
  - $\leadsto$ as soon as we have a little bit more than CC
  - $\leadsto$ we need a more computation-relevant presentation of CC

Thanks for your attention.