Dialectica the Ultimate

Pierre-Marie Pédrot

INRIA

TLLA'24
08/07/24

In Media Res

Today's Menu

The most tricky question that I have been tormented with for a while

In Media Res

Today's Menu

The most tricky question that I have been tormented with for a while

« What the (He)LL am I am going to talk about at TLLA? »

In Media Res

Today's Menu

The most tricky question that I have been tormented with for a while

« What the (He)LL am I am going to talk about at TLLA? »

Just in case you entered this room by sheer mistake:

TLLA :≡ Trends in **Linear Logic** and Applications

In Media Res

Today's Menu

The most tricky question that I have been tormented with for a while

« What the (He)LL am I am going to talk about at TLLA? »

Just in case you entered this room by sheer mistake:

$$\text{TLLA} :\equiv \text{Trends in } \textbf{Linear Logic} \text{ and Applications}$$

(It is still time to escape.)

How do you do, fellow linear logicians?

Linear Logic?

- LL was fashionable as an undergraduate
- We had a group discussing the latest GoI trends and stuff
- I betrayed linear logic right during my PhD (defended in 2015)
- I *think* I have been mostly doing dependent type theory since then

How do you do, fellow linear logicians?

Linear Logic?

- LL was fashionable as an undergraduate
- We had a group discussing the latest GoI trends and stuff
- I betrayed linear logic right during my PhD (defended in 2015)
- I *think* I have been mostly doing dependent type theory since then

How do you do, fellow linear logicians?

Linear Logic?

- LL was fashionable as an undergraduate
- We had a group discussing the latest GoI trends and stuff
- I betrayed linear logic right during my PhD (defended in 2015)
- I *think* I have been mostly doing dependent type theory since then

« Seriously, what I am going to talk about at TLLA? »

How I Learned to Stop Worrying and Love the Bang

The universal solution to the TLLA problem

(Hint: it's actually in the title.)

How I Learned to Stop Worrying and Love the Bang

The universal solution to the TLLA problem

(Hint: it's actually in the title.)

Dialectica.

How I Learned to Stop Worrying and Love the Bang

The universal solution to the TLLA problem

(Hint: it's actually in the title.)

Dialectica.

- Shameless recycling: this was the topic of my PhD

How I Learned to Stop Worrying and Love the Bang

The universal solution to the TLLA problem

(Hint: it's actually in the title.)

Dialectica.

- Shameless recycling: this was the topic of my PhD
- Advertisement-worthy: it is not taught around enough to my taste

How I Learned to Stop Worrying and Love the Bang

The universal solution to the TLLA problem

(Hint: it's actually in the title.)

Dialectica.

- Shameless recycling: this was the topic of my PhD
- Advertisement-worthy: it is not taught around enough to my taste
- Latent schizophrenia: it keeps appearing everywhere (and nobody sees it)

A Short, Mostly Wrong History of Dialectica

Dialectica in a nutshell

- The great ancestor of realizability
- The name comes from the journal it was published in
- Basically a fancy realizability model of HA^ω into System T
- Designed by Gödel in the 30's but published in 1958
- *Extremely* kludgy and antiquated

A Short, Mostly Wrong History of Dialectica

Dialectica in a nutshell

- The great ancestor of realizability
- The name comes from the journal it was published in
- Basically a fancy realizability model of HA^ω into System T
- Designed by Gödel in the 30's but published in 1958
- *Extremely* kludgy and antiquated

The More You Know

In *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes*, Gödel passes more time discussing α-conversion than the actual Dialectica interpretation.

A Short, Mostly Wrong History of Dialectica

Dialectica in a nutshell

- The great ancestor of realizability
- The name comes from the journal it was published in
- Basically a fancy realizability model of HA^ω into System T
- Designed by Gödel in the 30's but published in 1958
- *Extremely* kludgy and antiquated

The More You Know

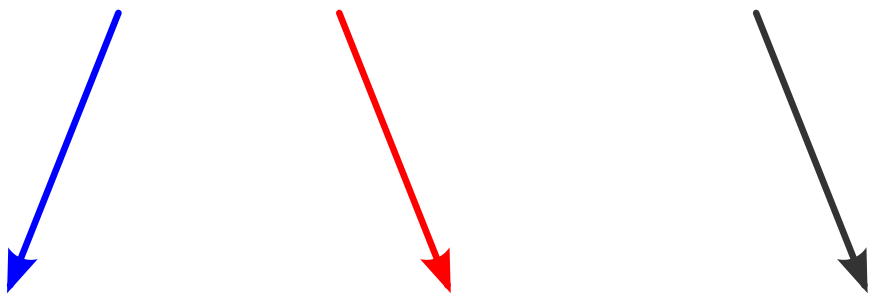
In *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes*, Gödel passes more time discussing α-conversion than the actual Dialectica interpretation.

(You have been warned.)

Anatomy of a Dusty Realizability

If $\mathsf{HA}^\omega \vdash A$ then

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



$\mathbb{W}(A)$	$\mathbb{C}(A)$	$\bot_A$
Sequence of types	Sequence of types	$\mathbb{W}(A) \to \mathbb{C}(A) \to \mathtt{Prop}$
Witness	Counter	Orthogonality

We Have Come to Realize

Legitimate question

What has this to do with LL?

We Have Come to Realize

Legitimate question

What has this to do with LL?

- On most connectives, Dialectica follows the BHK interpretation.

$$\begin{aligned} \mathbb{W}(A \wedge B) &= \mathbb{W}(A) \times \mathbb{W}(B) \\ \mathbb{W}(A \vee B) &= \mathbb{W}(A) + \mathbb{W}(B) \end{aligned}$$

We Have Come to Realize

Legitimate question

What has this to do with LL?

- On most connectives, Dialectica follows the BHK interpretation.

$$\begin{aligned} \mathbb{W}(A \wedge B) &= \mathbb{W}(A) \times \mathbb{W}(B) \\ \mathbb{W}(A \vee B) &= \mathbb{W}(A) + \mathbb{W}(B) \end{aligned}$$

- That is, it is morally the same as Kreisel realizability
- i.e. boring intuitionistic semantics

We Have Come to Realize

Legitimate question

What has this to do with LL?

- On most connectives, Dialectica follows the BHK interpretation.

$$\begin{aligned} \mathbb{W}(A \wedge B) &= \mathbb{W}(A) \times \mathbb{W}(B) \\ \mathbb{W}(A \vee B) &= \mathbb{W}(A) + \mathbb{W}(B) \end{aligned}$$

- That is, it is morally the same as Kreisel realizability
- i.e. boring intuitionistic semantics

... except the one important connective

We Have Come to Realize

Legitimate question

What has this to do with LL?

- On most connectives, Dialectica follows the BHK interpretation.

$$\begin{aligned} \mathbb{W}(A \wedge B) &= \mathbb{W}(A) \times \mathbb{W}(B) \\ \mathbb{W}(A \vee B) &= \mathbb{W}(A) + \mathbb{W}(B) \end{aligned}$$

- That is, it is morally the same as Kreisel realizability
- i.e. boring intuitionistic semantics

... except the one important connective

$$A \to B$$

Implying this Makes Sense

There is more to arrows than just functions!

$$\mathbb{W}(A \to B) = \begin{cases} \mathbb{W}(A) \Rightarrow \mathbb{W}(B) \\ \mathbb{W}(A) \Rightarrow \mathbb{C}(B) \Rightarrow \mathbb{C}(A) \end{cases}$$

$$\mathbb{C}(A \to B) = \mathbb{W}(A) \times \mathbb{C}(B)$$

Functions have both a **forward** and a **backward** component.

Implying this Makes Sense

There is more to arrows than just functions!

$$\mathbb{W}(A \to B) \quad = \quad \begin{cases} \mathbb{W}(A) \Rightarrow \mathbb{W}(B) \\ \mathbb{W}(A) \Rightarrow \mathbb{C}(B) \Rightarrow \mathbb{C}(A) \end{cases}$$

$$\mathbb{C}(A \to B) \quad = \quad \mathbb{W}(A) \times \mathbb{C}(B)$$

Functions have both a **forward** and a **backward** component.

A clear reminiscence of LL!

$$!A \multimap B \quad \cong \quad B^{\perp} \multimap ?A^{\perp}$$

Beating a Dead Horse

Since De Paiva, it is well-known that Dialectica factorizes through LL

$$
\begin{aligned}
\mathbb{W}(A \multimap B) &= \begin{cases} \mathbb{W}(A) \Rightarrow \mathbb{W}(B) \\ \mathbb{C}(B) \Rightarrow \mathbb{C}(A) \end{cases} \\
\mathbb{C}(A \multimap B) &= \mathbb{W}(A) \times \mathbb{C}(B) \\
\\
\mathbb{W}(!A) &= \mathbb{W}(A) \\
\mathbb{C}(!A) &= \mathbb{W}(A) \Rightarrow \mathbb{C}(A)
\end{aligned}
$$

Beating a Dead Horse

Since De Paiva, it is well-known that Dialectica factorizes through LL

$$\begin{aligned}
\mathbb{W}(A \multimap B) &= \begin{cases} \mathbb{W}(A) \Rightarrow \mathbb{W}(B) \\ \mathbb{C}(B) \Rightarrow \mathbb{C}(A) \end{cases} \\
\mathbb{C}(A \multimap B) &= \mathbb{W}(A) \times \mathbb{C}(B)
\end{aligned}$$

$$\begin{aligned}
\mathbb{W}(!A) &= \mathbb{W}(A) \\
\mathbb{C}(!A) &= \mathbb{W}(A) \Rightarrow \mathbb{C}(A)
\end{aligned}$$

$$\mathbb{W}(A \to B) \cong \mathbb{W}(!A \multimap B)$$

Reminder:

$$\mathbb{W}(A \to B) = \begin{cases} \mathbb{W}(A) \Rightarrow \mathbb{W}(B) \\ \mathbb{W}(A) \Rightarrow \mathbb{C}(B) \Rightarrow \mathbb{C}(A) \end{cases}$$

Can Dialectica Break Bricks?

Actually, most* models of LL are Dialectica-like

- Chu spaces
- Dialectica categories
- Double-glueing

Can Dialectica Break Bricks?

Actually, most* models of LL are Dialectica-like

- Chu spaces
- Dialectica categories
- Double-glueing

Nothing new here, nonetheless, it does not hurt to stress it.

Can Dialectica Break Bricks?

Actually, most* models of LL are Dialectica-like

- Chu spaces
- Dialectica categories
- Double-glueing

Nothing new here, nonetheless, it does not hurt to stress it.

Empirical Observation

Dialectica is the most natural way to linearize an intuitionistic calculus

Can Dialectica Break Bricks?

Actually, most* models of LL are Dialectica-like

- Chu spaces
- Dialectica categories
- Double-glueing

Nothing new here, nonetheless, it does not hurt to stress it.

Empirical Observation

Dialectica is the most natural way to linearize an intuitionistic calculus

(∗) except maybe games, and even there if we squint enough there are weird similarities.

The Horse that Did Not Stop Beating Itself to Death

Dialectica keeps popping everywhere

The Horse that Did Not Stop Beating Itself to Death

Dialectica keeps popping everywhere

- Some of these Dialectica-like constructs are well-known
- Some are more obscure, e.g. sequential algorithms (cf. some workshop in Marseille)
- Surprisingly, **obvious** *but* **deep** connections were only made recently

The Horse that Did Not Stop Beating Itself to Death

Dialectica keeps popping everywhere

- Some of these Dialectica-like constructs are well-known
- Some are more obscure, e.g. sequential algorithms (cf. some workshop in Marseille)
- Surprisingly, **obvious** *but* **deep** connections were only made recently

Infomercial

This afternoon at LICS: Dialectica actually computes differentials

The Horse that Did Not Stop Beating Itself to Death

Dialectica keeps popping everywhere

- Some of these Dialectica-like constructs are well-known
- Some are more obscure, e.g. sequential algorithms (cf. some workshop in Marseille)
- Surprisingly, **obvious** *but* **deep** connections were only made recently

Infomercial

This afternoon at LICS: Dialectica actually computes differentials

My now systematic reaction when faced with something related to LL

In This Talk

A 100% free idea for a research project!

In This Talk

A 100% free idea for a research project!

- I am too lazy to write a paper on that
- It is more efficient to point at the moon
- Invited talks are great for dissemination of such knowledge

In This Talk

A 100% free idea for a research project!

- I am too lazy to write a paper on that
- It is more efficient to point at the moon
- Invited talks are great for dissemination of such knowledge

Graded / quantitative types are a poor man's Dialectica

In This Talk

A 100% free idea for a research project!

- I am too lazy to write a paper on that
- It is more efficient to point at the moon
- Invited talks are great for dissemination of such knowledge

Graded / quantitative types are a poor man's Dialectica

- More positively, Dialectica is a finer kind of graded types
- Compatible with rich types (i.e. MLTT)
- Dialectica as proof-relevant, higher-order complexity annotations

Part I
Dialectica Done Right

Curry-Howard to the Rescue

Let us present Dialectica as a program translation!

- Without cheating too much
- A rich language in the source: MLTT
- The target will also be a variant of MLTT (some cheating involved here)
- A variant of Moss-von Glehn model (a.k.a the Google Drive note by A. Bauer and myself)

Curry-Howard to the Rescue

Let us present Dialectica as a program translation!

- Without cheating too much
- A rich language in the source: MLTT
- The target will also be a variant of MLTT (some cheating involved here)
- A variant of Moss-von Glehn model (a.k.a the Google Drive note by A. Bauer and myself)

... but *why*?

Curry-Howard to the Rescue

Let us present Dialectica as a program translation!

- Without cheating too much
- A rich language in the source: MLTT
- The target will also be a variant of MLTT (some cheating involved here)
- A variant of Moss-von Glehn model (a.k.a the Google Drive note by A. Bauer and myself)

... but *why*?

- To get rid of most of Gödel's hacks
- To care about the equational theory
- Because dependent types are cool, expressive and effective

MLTT for Dummies

Do not panic

I am not expecting you to be experts in dependent types

$\rightsquigarrow$ MLTT is just a fancily-typed λ-calculus. (Any Agda or Coq user around?)

MLTT for Dummies

Do not panic

I am not expecting you to be experts in dependent types

$\rightsquigarrow$ MLTT is just a fancily-typed λ-calculus. (Any Agda or Coq user around?)

Otherwise

The major differences with your run-of-the-mill λ-calculus.

- Types can depend on terms (duh!)

$$\Pi(x : A).\, B$$

- The equational theory on terms lifts to the type layer

$$\texttt{refl} : \Pi(A : \texttt{Type})(x : A).\, x = x \qquad \rightsquigarrow \qquad (\texttt{refl}\ \mathbb{N}\ 2) : 1 + 1 = 2$$

- Large elimination: build types by pattern-matching on positives

$$\lambda(b : \mathbb{B}).\, \texttt{if}\ b\ \texttt{then}\ \mathbb{N}\ \texttt{else}\ (\mathbb{N} \to \mathbb{N}) \quad : \quad \mathbb{B} \to \texttt{Type}$$

Simpler? It Depends

Remember that in standard Dialectica, a type A is converted to

a type $\mathbb{W}(A)$ a type $\mathbb{C}(A)$ a relation $\perp_A \subseteq \mathbb{W}(A) \times \mathbb{C}(A)$

Simpler? It Depends

Remember that in standard Dialectica, a type A is converted to

$$\text{a type } \mathbb{W}(A) \quad \text{a type } \mathbb{C}(A) \quad \text{a relation } \perp_A \subseteq \mathbb{W}(A) \times \mathbb{C}(A)$$

Thanks to the added expressivity, we can conflate $\mathbb{C}(A)$ and $\perp_A$

An MLTT type A will be converted to

- a type $\mathbb{W}(A)$
- a dependent type $\mathbb{C}(A) : \mathbb{W}(A) \to \mathtt{Type}$

No need for orthogonality, it is built into $\mathbb{C}$!

Simpler? It Depends

Remember that in standard Dialectica, a type A is converted to

$$\text{a type } \mathbb{W}(A) \quad \text{a type } \mathbb{C}(A) \quad \text{a relation } \perp_A \subseteq \mathbb{W}(A) \times \mathbb{C}(A)$$

Thanks to the added expressivity, we can conflate $\mathbb{C}(A)$ and $\perp_A$

An MLTT type A will be converted to

- a type $\mathbb{W}(A)$
- a dependent type $\mathbb{C}(A) : \mathbb{W}(A) \to \mathtt{Type}$

No need for orthogonality, it is built into $\mathbb{C}$!

Remark: for readability I will write $\mathbb{C}(A)\langle M \rangle$ for $\mathbb{C}(A)\ M$.

Exponentially More Complex

We must validate the equational theory of MLTT

- in particular the β-rule (i.e. we want a CCC)

Exponentially More Complex

We must validate the equational theory of MLTT

- in particular the β-rule (i.e. we want a CCC)
- ... but the natural model arising from Dialectica is only a SMCC
- the original Dialectica is *not* a program translation

Exponentially More Complex

We must validate the equational theory of MLTT

- in particular the β-rule (i.e. we want a CCC)
- ... but the natural model arising from Dialectica is only a SMCC
- the original Dialectica is *not* a program translation

We must introduce an additional structure $\mathfrak{M}$ called **abstract multisets**

- A generalization of finite multisets
- Clearly there to handle exponentials
- Bear with me...

Categorical Nonsense

An abstract multiset structure is just a semiringoid!

- A monad $\mathfrak{M}$ with return and bind:

$$\{\cdot\} : A \to \mathfrak{M}\ A \qquad \gg\!= : \mathfrak{M}\ A \to (A \to \mathfrak{M}\ B) \to \mathfrak{M}\ B$$

- that further has a commutative monoid structure:

$$\varnothing : \mathfrak{M}\ A \qquad \oplus : \mathfrak{M}\ A \to \mathfrak{M}\ A \to \mathfrak{M}\ A$$

Categorical Nonsense

An abstract multiset structure is just a semiringoid!

- A monad $\mathfrak{M}$ with return and bind:

$$\{\cdot\} : A \to \mathfrak{M}\ A \qquad \gg\!= : \mathfrak{M}\ A \to (A \to \mathfrak{M}\ B) \to \mathfrak{M}\ B$$

- that further has a commutative monoid structure:

$$\varnothing : \mathfrak{M}\ A \qquad \oplus : \mathfrak{M}\ A \to \mathfrak{M}\ A \to \mathfrak{M}\ A$$

If you forget the parameter, equations are those of a semiring

$$\{M\} \gg\!= F \equiv F\ M \qquad M \gg\!= (\lambda x.\, \{x\}) \equiv M$$

$$(M \gg\!= F) \gg\!= G \equiv M \gg\!= (\lambda x.\, F\ x \gg\!= G)$$

$$\varnothing \oplus M \equiv M \qquad M \oplus \varnothing \equiv M \qquad (M \oplus N) \oplus P \equiv M \oplus (N \oplus P)$$

$$\varnothing \gg\!= F \equiv \varnothing \qquad M \gg\!= (\lambda x.\, \varnothing) \equiv \varnothing$$

$$(M \oplus N) \gg\!= F \equiv (M \gg\!= F) \oplus (N \gg\!= F) \qquad M \gg\!= (\lambda x.\, F \oplus G) \equiv (M \gg\!= F) \oplus (M \gg\!= G)$$

Categorical Nonsense

An abstract multiset structure is just a semiringoid!

- A monad $\mathfrak{M}$ with return and bind:

$$\{\cdot\} : A \to \mathfrak{M}\ A \qquad \gg\!= : \mathfrak{M}\ A \to (A \to \mathfrak{M}\ B) \to \mathfrak{M}\ B$$

- that further has a commutative monoid structure:

$$\varnothing : \mathfrak{M}\ A \qquad \oplus : \mathfrak{M}\ A \to \mathfrak{M}\ A \to \mathfrak{M}\ A$$

If you forget the parameter, equations are those of a semiring

$$\{M\} \gg\!= F \equiv F\ M \qquad M \gg\!= (\lambda x.\, \{x\}) \equiv M$$

$$(M \gg\!= F) \gg\!= G \equiv M \gg\!= (\lambda x.\, F\ x \gg\!= G)$$

$$\varnothing \oplus M \equiv M \qquad M \oplus \varnothing \equiv M \qquad (M \oplus N) \oplus P \equiv M \oplus (N \oplus P)$$

$$\varnothing \gg\!= F \equiv \varnothing \qquad M \gg\!= (\lambda x.\, \varnothing) \equiv \varnothing$$

$$(M \oplus N) \gg\!= F \equiv (M \gg\!= F) \oplus (N \gg\!= F) \qquad M \gg\!= (\lambda x.\, F \oplus G) \equiv (M \gg\!= F) \oplus (M \gg\!= G)$$

Prototypical example: finite multisets

The Translation, At Last

Assume $\Gamma \vdash M : A$ in MLTT.

The Translation, At Last

Assume $\Gamma \vdash M : A$ in MLTT.

The Dialectica translation will produce **two** kinds of objects.

The Translation, At Last

Assume $\Gamma \vdash M : A$ in MLTT.

The Dialectica translation will produce **two** kinds of objects.

The **forward** translation:

$$\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$$

The Translation, At Last

Assume $\Gamma \vdash M : A$ in MLTT.

The Dialectica translation will produce **two** kinds of objects.

The **forward** translation:

$$\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$$

For **each** $x : X \in \Gamma$, a **reverse** translation:

$$\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

Reminder:

$$\mathfrak{M} : \mathtt{Type} \to \mathtt{Type} \qquad \mathbb{W}(A) : \mathtt{Type} \qquad \mathbb{C}(A) : \mathbb{W}(A) \to \mathtt{Type}$$

The Translation, At Last

Assume $\Gamma \vdash M : A$ in MLTT.

The Dialectica translation will produce **two** kinds of objects.

The **forward** translation:

$$\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$$

For **each** $x : X \in \Gamma$, a **reverse** translation:

$$\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

Reminder:

$$\mathfrak{M} : \mathtt{Type} \to \mathtt{Type} \qquad \mathbb{W}(A) : \mathtt{Type} \qquad \mathbb{C}(A) : \mathbb{W}(A) \to \mathtt{Type}$$

If $\|\Gamma\| = n$, this means I have $n + 1$ objects!

Motto

$[M]_x$ is a measure of the "number" of uses of x in M

$$\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

Motto

$[M]_x$ is a measure of the "number" of uses of x in M

$$\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

In particular,

- $[x]_x := \lambda\pi.\,\{\pi\}$ (dereliction)
- $[y]_x := \lambda\pi.\,\varnothing$ if $x \neq y$ (weakening)

noting that

$$[x] := x$$

(Reminder: $\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$)

Motto

$[M]_x$ is a measure of the "number" of uses of x in M

$$\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

In particular,

- $[x]_x := \lambda\pi.\,\{\pi\}$ (dereliction)
- $[y]_x := \lambda\pi.\,\varnothing$ if $x \neq y$ (weakening)

noting that

$$[x] := x$$

(Reminder: $\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$)

Substitution lemma (contraction + promotion)

$$[M\{x := N\}]_y\ \pi \quad \equiv \quad ([M]_y\{x := [N]\}\ \pi) \ \oplus\ ([M]_x\{x := [N]\}\ \pi \gg\!= [N]_y)$$

A Functional Functional Interpretation

There is no structure in our theory yet, let's look at functions!

$$\mathbb{W}(\Pi(x : A).\, B) \qquad := \left\{ \begin{array}{l} f : \Pi(x : \mathbb{W}(A)).\, \mathbb{W}(B) \\ \varphi : \Pi(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle \to \mathfrak{M}\ \mathbb{C}(A)\langle x\rangle \end{array} \right\}$$

$$\mathbb{C}(\Pi(x : A).\, B)\langle f, \varphi\rangle := \Sigma(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle$$

A Functional Functional Interpretation

There is no structure in our theory yet, let's look at functions!

$$\mathbb{W}(\Pi(x : A).\, B) \qquad := \left\{ \begin{array}{l} f : \Pi(x : \mathbb{W}(A)).\, \mathbb{W}(B) \\ \varphi : \Pi(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle \to \mathfrak{M}\ \mathbb{C}(A)\langle x\rangle \end{array} \right\}$$

$$\mathbb{C}(\Pi(x : A).\, B)\langle f, \varphi\rangle := \Sigma(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle$$

Up to dependent noise, this is the same as the simply typed variant!

A Functional Functional Interpretation

There is no structure in our theory yet, let's look at functions!

$$\mathbb{W}(\Pi(x : A).\, B) \quad := \left\{ \begin{array}{l} f : \Pi(x : \mathbb{W}(A)).\, \mathbb{W}(B) \\ \varphi : \Pi(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle \to \mathfrak{M}\ \mathbb{C}(A)\langle x\rangle \end{array} \right\}$$

$$\mathbb{C}(\Pi(x : A).\, B)\langle f, \varphi\rangle := \Sigma(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x\rangle$$

Up to dependent noise, this is the same as the simply typed variant!

Forward translations are easy.

$$\begin{array}{lcl} [\lambda x.\, M] & := & (\lambda x.\, [M]), (\lambda x.\, [M]_x) \\ [M\ N] & := & [M].1\ [N] \end{array}$$

Recall: if $\Gamma \vdash M : A$ then $\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M]\rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x\rangle$ provided $x : X \in \Gamma$

Full Reverse

Reverse translations are quite a mouthful.

$$[\lambda x.\, M]_x := \lambda(x, \pi).\, [M]_x\ \pi$$

$$[M\ N]_x := \lambda\pi.\quad ([M].2\ [N]\ \pi)\ \oplus\ ([M]_x\ ([N], \pi) \gg\!= [N]_y)$$

Cheat Sheet

1. If $\Gamma \vdash M : A$ then $\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$ provided $x : X \in \Gamma$.
2. $\mathbb{W}(\Pi(x : A).\, B) := \left\{ \begin{array}{l} f : \Pi(x : \mathbb{W}(A)).\, \mathbb{W}(B) \\ \varphi : \Pi(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x \rangle \to \mathfrak{M}\ \mathbb{C}(A)\langle x \rangle \end{array} \right\}$
3. $\mathbb{C}(\Pi(x : A).\, B)\langle f, \varphi \rangle := \Sigma(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x \rangle$

Full Reverse

Reverse translations are quite a mouthful.

$$[\lambda x.\, M]_x := \lambda(x, \pi).\, [M]_x\ \pi$$

$$[M\ N]_x := \lambda\pi.\quad ([M].2\ [N]\ \pi)\ \oplus\ ([M]_x\ ([N], \pi) \gg\!= [N]_y)$$

Cheat Sheet

1. If $\Gamma \vdash M : A$ then $\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$ provided $x : X \in \Gamma$.
2. $\mathbb{W}(\Pi(x : A).\, B) := \left\{ \begin{array}{l} f : \Pi(x : \mathbb{W}(A)).\, \mathbb{W}(B) \\ \varphi : \Pi(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x \rangle \to \mathfrak{M}\ \mathbb{C}(A)\langle x \rangle \end{array} \right\}$
3. $\mathbb{C}(\Pi(x : A).\, B)\langle f, \varphi \rangle := \Sigma(x : \mathbb{W}(A)).\, \mathbb{C}(B)\langle f\ x \rangle$

Application basically follows the substitution lemma.

Positive Attitude

I have been bluffing you: a dependent $\mathbb{C}$ is only needed for positive types.

$\rightsquigarrow$ most of the dependent noise so far has been useless.

Positive Attitude

I have been bluffing you: a dependent $\mathbb{C}$ is only needed for positive types.

$\rightsquigarrow$ most of the dependent noise so far has been useless.

I don't want to enter details, here is a quick sketch of sum types:

$$\mathbb{W}(A + B) \qquad := \quad \mathbb{W}(A) + \mathbb{W}(B)$$

Positive Attitude

I have been bluffing you: a dependent $\mathbb{C}$ is only needed for positive types.

$\rightsquigarrow$ most of the dependent noise so far has been useless.

I don't want to enter details, here is a quick sketch of sum types:

$$\begin{array}{lcl} \mathbb{W}(A + B) & := & \mathbb{W}(A) + \mathbb{W}(B) \\ \mathbb{C}(A + B)\langle \texttt{inl}\ M \rangle & := & \mathbb{C}(A)\langle M \rangle \\ \mathbb{C}(A + B)\langle \texttt{inr}\ N \rangle & := & \mathbb{C}(B)\langle N \rangle \end{array}$$

Positive Attitude

I have been bluffing you: a dependent $\mathbb{C}$ is only needed for positive types.

$\rightsquigarrow$ most of the dependent noise so far has been useless.

I don't want to enter details, here is a quick sketch of sum types:

$$\begin{array}{lcl} \mathbb{W}(A + B) & := & \mathbb{W}(A) + \mathbb{W}(B) \\ \mathbb{C}(A + B)\langle \texttt{inl}\ M\rangle & := & \mathbb{C}(A)\langle M\rangle \\ \mathbb{C}(A + B)\langle \texttt{inr}\ N\rangle & := & \mathbb{C}(B)\langle N\rangle \end{array}$$

$\rightsquigarrow$ term translation follows the leading motto

- Variable use in $\texttt{inl}\ M$ and $\texttt{inr}\ M$ is obvious
- Variable use in

$$\texttt{case}\ M\ \texttt{with}\ \texttt{inl}\ x \to N_1 \mid \texttt{inr}\ y \to N_2$$

comes from either M or N_i depending on M

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

$$\begin{array}{lcl} \mathbb{W}(\mathtt{Type}) & := & \{\ \mathbf{W} : \mathtt{Type};\ \mathbf{C} : \mathbf{W} \to \mathtt{Type}\ \} \\ \mathbb{C}(\mathtt{Type})\langle \mathbf{W}, \mathbf{C} \rangle & := & \text{whatever} \end{array}$$

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

$$\begin{array}{lcl} \mathbb{W}(\mathtt{Type}) & := & \{\ \mathbf{W} : \mathtt{Type};\ \mathbf{C} : \mathbf{W} \to \mathtt{Type}\ \} \\ \mathbb{C}(\mathtt{Type})\langle \mathbf{W}, \mathbf{C} \rangle & := & \text{whatever} \end{array}$$

$$\mathbb{W}(A) := [A].\mathbf{W} \qquad \mathbb{C}(A) := [A].\mathbf{C}$$

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

$$\begin{array}{lcl} \mathbb{W}(\mathtt{Type}) & := & \{\ \mathbf{W} : \mathtt{Type};\ \mathbf{C} : \mathbf{W} \to \mathtt{Type}\ \} \\ \mathbb{C}(\mathtt{Type})\langle \mathbf{W}, \mathbf{C} \rangle & := & \text{whatever} \end{array}$$

$$\mathbb{W}(A) := [A].\mathbf{W} \qquad \mathbb{C}(A) := [A].\mathbf{C}$$

$\rightsquigarrow$ for the forward translation of a type A we have little choice

$$[A] := (\mathbb{W}(A), \mathbb{C}(A))$$

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

$$\begin{array}{lcl} \mathbb{W}(\mathtt{Type}) & := & \{\ \mathbf{W} : \mathtt{Type};\ \mathbf{C} : \mathbf{W} \to \mathtt{Type}\ \} \\ \mathbb{C}(\mathtt{Type})\langle \mathbf{W}, \mathbf{C} \rangle & := & \text{whatever} \end{array}$$

$$\mathbb{W}(A) := [A].\mathbf{W} \qquad \mathbb{C}(A) := [A].\mathbf{C}$$

$\rightsquigarrow$ for the forward translation of a type A we have little choice

$$[A] := (\mathbb{W}(A), \mathbb{C}(A))$$

$\rightsquigarrow$ for the reverse translation, no constraints ("types do not use resources")

$$[A]_x := \lambda\pi.\varnothing$$

The Universal Property of Universes

As usual in effectful models of MLTT, types-as-terms are underspecified

$$\begin{array}{lcl} \mathbb{W}(\mathtt{Type}) & := & \{\ \mathbf{W} : \mathtt{Type};\ \mathbf{C} : \mathbf{W} \to \mathtt{Type}\ \} \\ \mathbb{C}(\mathtt{Type})\langle \mathbf{W}, \mathbf{C} \rangle & := & \text{whatever} \end{array}$$

$$\mathbb{W}(A) := [A].\mathbf{W} \qquad \mathbb{C}(A) := [A].\mathbf{C}$$

$\rightsquigarrow$ for the forward translation of a type A we have little choice

$$[A] := (\mathbb{W}(A), \mathbb{C}(A))$$

$\rightsquigarrow$ for the reverse translation, no constraints ("types do not use resources")

$$[A]_x := \lambda\pi.\varnothing$$

A lot of somewhat arbitrary choices

Trust me

We have defined a model of MLTT.

If $\Gamma \vdash M : A$,

- $\mathbb{W}(\Gamma) \vdash [M] : \mathbb{W}(A)$
- $\mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$ for all $x : X \in \Gamma$

and similarly for conversion.

Part II

A Resourceful Take

People in type theory want to track resources

- for efficiency
- for security / privacy
- for the love of LL

A Resourceful Take

People in type theory want to track resources

- for efficiency
- for security / privacy
- for the love of LL

Various names for somewhat similar systems

- Graded type theories (à la Orchard)
- Quantitative type theories (à la McBride-Atkey)

A Resourceful Take

People in type theory want to track resources

- for efficiency
- for security / privacy
- for the love of LL

Various names for somewhat similar systems

- Graded type theories (à la Orchard)
- Quantitative type theories (à la McBride-Atkey)

As a first-order approximation I will lump them together here

- I will gloss over the differences
- and deliberately ignore some technicalities

Get in the Ring

Typically, fix a **resource** semi-ring $(\mathbb{M}, +, \times)$.

$\rightsquigarrow$ used to assign a measure to variable use

- 0 means not used
- 1 is exactly once

Get in the Ring

Typically, fix a **resource** semi-ring $(\mathbb{M}, +, \times)$.

$\rightsquigarrow$ used to assign a measure to variable use

- 0 means not used
- 1 is exactly once

$\rightsquigarrow$ you tend to expect a little bit more structure

- an order $\leq$ compatible with the semi-ring operations
- often a maximal absorbing element ω
- sometimes a max operation $\sqcup$

Get in the Ring

Typically, fix a **resource** semi-ring $(\mathbb{M}, +, \times)$.

$\rightsquigarrow$ used to assign a measure to variable use

- 0 means not used
- 1 is exactly once

$\rightsquigarrow$ you tend to expect a little bit more structure

- an order $\leq$ compatible with the semi-ring operations
- often a maximal absorbing element ω
- sometimes a max operation $\sqcup$

Simplest example is $0 \leq 1 \leq \omega$.

Varying Variables

Variables are annotated by the semi-ring

$$x_1 :_{\alpha_1} X_1, \ldots x :_{\alpha_n} X_n \vdash M : A$$

where $\alpha_i : \mathbb{M}$

Varying Variables

Variables are annotated by the semi-ring

$$x_1 :_{\alpha_1} X_1, \ldots x :_{\alpha_n} X_n \vdash M : A$$

where $\alpha_i : \mathbb{M}$

$\rightsquigarrow$ the semi-ring structure lifts to contexts naturally.

Varying Variables

Variables are annotated by the semi-ring

$$x_1 :_{\alpha_1} X_1, \ldots x :_{\alpha_n} X_n \vdash M : A$$

where $\alpha_i : \mathbb{M}$

$\rightsquigarrow$ the semi-ring structure lifts to contexts naturally.

Once again, more subtleties in practice

- GrTT has a separate annotation for term / type usage
- QTT also annotates the sequent itself

Varying Variables

Variables are annotated by the semi-ring

$$x_1 :_{\alpha_1} X_1, \ldots x :_{\alpha_n} X_n \vdash M : A$$

where $\alpha_i : \mathbb{M}$

$\rightsquigarrow$ the semi-ring structure lifts to contexts naturally.

Once again, more subtleties in practice

- GrTT has a separate annotation for term / type usage
- QTT also annotates the sequent itself

We should concentrate on the big picture!

Grading Types

Types and terms are annotated accordingly

$$A, B ::= \ldots \mid \Pi(x :_\alpha A).\, B \mid \ldots$$

Grading Types

Types and terms are annotated accordingly

$$A, B ::= \ldots \mid \Pi(x :_\alpha A).\, B \mid \ldots$$

Some typing rules are expected

$$\frac{}{0\Gamma, x :_1 A \vdash x : A}$$

$$\frac{\Gamma, x :_\alpha A \vdash M : B}{\Gamma \vdash \lambda x.\, M : \Pi(x :_\alpha A).\, B}$$

Grading Types

Types and terms are annotated accordingly

$$A, B ::= \ldots \mid \Pi(x :_\alpha A).\, B \mid \ldots$$

Some typing rules are expected

$$\frac{}{0\Gamma, x :_1 A \vdash x : A}$$

$$\frac{\Gamma, x :_\alpha A \vdash M : B}{\Gamma \vdash \lambda x.\, M : \Pi(x :_\alpha A).\, B}$$

Sprinkle with subtyping induced by $\leq$

Apply Copiously

More interesting is the case of application!

$$\frac{\Gamma \sim \Delta \qquad \Gamma \vdash M : \Pi(x :_\alpha A).\, B \qquad \Delta \vdash N : A}{\Gamma + (\alpha \times \Delta) \vdash M\ N : B\{x := N\}}$$

where $\Gamma \sim \Delta$ means Γ and Δ are the same context up to annotations.

Apply Copiously

More interesting is the case of application!

$$\frac{\Gamma \sim \Delta \qquad \Gamma \vdash M : \Pi(x :_\alpha A).\, B \qquad \Delta \vdash N : A}{\Gamma + (\alpha \times \Delta) \vdash M\ N : B\{x := N\}}$$

where $\Gamma \sim \Delta$ means Γ and Δ are the same context up to annotations.

This is really where the semi-ring structure shines!

The Sad Truth

Why are graded types too limited?

The Sad Truth

Why are graded types too limited?

The semi-ring gives very little expressivity!

- Poor language of annotations
- Despairingly simple-typed
- Often needs to overapproximate
- Not very modular

The Sad Truth

Why are graded types too limited?

The semi-ring gives very little expressivity!

- Poor language of annotations
- Despairingly simple-typed
- Often needs to overapproximate
- Not very modular

Let me give some examples of increasing annoyance

There is no Alternative

How to type:

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2$$

There is no Alternative

How to type:

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2$$

Solution 1:

- ask that N_i use variables the same number of times
- ... and hope that subtyping will save you

There is no Alternative

How to type:

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2$$

Solution 1:

- ask that N_i use variables the same number of times
- ... and hope that subtyping will save you

Solution 2:

- assume finite semi-lattice structure
- claim that the variable uses in branches are the max from N_i

There is no Alternative

How to type:

$$\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2$$

Solution 1:

- ask that N_i use variables the same number of times
- ... and hope that subtyping will save you

Solution 2:

- assume finite semi-lattice structure
- claim that the variable uses in branches are the max from N_i

Both are overapproximations: it actually depends on M

$$\texttt{if } M \texttt{ then } (\texttt{if } M \texttt{ then } N_1 \texttt{ else } N_2) \texttt{ else } N_3$$

There is no Countable Alternative

This was the easy one! There are only two branches for $\mathbb{B}$.

And if we want to quantitatively type a recursion on $\mathbb{N}$?

There is no Countable Alternative

This was the easy one! There are only two branches for $\mathbb{B}$.

And if we want to quantitatively type a recursion on $\mathbb{N}$?

- Solution 1 works but it's a wild overapproximation
- Solution 2 requires the semiring to be a countably complete lattice

There is no Countable Alternative

This was the easy one! There are only two branches for $\mathbb{B}$.

And if we want to quantitatively type a recursion on $\mathbb{N}$?

- Solution 1 works but it's a wild overapproximation
- Solution 2 requires the semiring to be a countably complete lattice

Both solutions are quite bad.

I Want My Functions Back

It is actually worse: HO functions are basically broken

$$(A \to_\alpha B) \to_\beta C$$

I Want My Functions Back

It is actually worse: HO functions are basically broken

$$(A \to_\alpha B) \to_\beta C$$

The number of uses α is fixed once and for all

I Want My Functions Back

It is actually worse: HO functions are basically broken

$$(A \to_\alpha B) \to_\beta C$$

The number of uses α is fixed once and for all

- This completely prevents a fine analysis of its resources
- What happens when the number of calls is not static?

I Want My Functions Back

It is actually worse: HO functions are basically broken

$$(A \to_\alpha B) \to_\beta C$$

The number of uses α is fixed once and for all

- This completely prevents a fine analysis of its resources
- What happens when the number of calls is not static?

What happens in practice: (a.k.a. fancy subtyping doesn't work)

- $0 \rightsquigarrow$ runtime irrelevant terms (e.g. types)
- $1 \rightsquigarrow$ extremely rare cases (e.g. linear state-passing)
- $\omega \rightsquigarrow$ all other cases

I Want My Functions Back

It is actually worse: HO functions are basically broken

$$(A \to_\alpha B) \to_\beta C$$

The number of uses α is fixed once and for all

- This completely prevents a fine analysis of its resources
- What happens when the number of calls is not static?

What happens in practice: (a.k.a. fancy subtyping doesn't work)

- $0 \rightsquigarrow$ runtime irrelevant terms (e.g. types)
- $1 \rightsquigarrow$ extremely rare cases (e.g. linear state-passing)
- $\omega \rightsquigarrow$ all other cases

You have just reinvented a single runtime irrelevant modality

Obvious Progaganda is Obvious

To be fair, there have been proposals to extend the expressivity

... but in my opinion these are very contrived.

Obvious Progaganda is Obvious

To be fair, there have been proposals to extend the expressivity

... but in my opinion these are very contrived.

What if I told you

we already have a solution?

Obvious Progaganda is Obvious

To be fair, there have been proposals to extend the expressivity

... but in my opinion these are very contrived.

What if I told you

we already have a solution?

DIALECTICA

(what a surprise!)

The Main Idea

The semi-ring annotations should depend on terms

This is literally the only reason why we had to overapproximate.

The Main Idea

The semi-ring annotations should depend on terms

This is literally the only reason why we had to overapproximate.

- A function should tell you **dynamically** its argument measure
- Pattern-matching measure should depend on the scrutinee
- This clearly requires dependent types

The Main Idea

The semi-ring annotations should depend on terms

This is literally the only reason why we had to overapproximate.

- A function should tell you **dynamically** its argument measure
- Pattern-matching measure should depend on the scrutinee
- This clearly requires dependent types

For this, we need to turn $x :_\alpha X$ into something that looks like

$$\Gamma \vdash M : A \quad \text{and} \quad x : X \in \Gamma \quad \rightsquigarrow \quad \alpha_x(M) : \mathcal{O}_A^\Gamma(X) \to \mathbb{M}$$

The Main Idea

The semi-ring annotations should depend on terms

This is literally the only reason why we had to overapproximate.

- A function should tell you **dynamically** its argument measure
- Pattern-matching measure should depend on the scrutinee
- This clearly requires dependent types

For this, we need to turn $x :_\alpha X$ into something that looks like

$$\Gamma \vdash M : A \quad \text{and} \quad x : X \in \Gamma \quad \rightsquigarrow \quad \alpha_x(M) : \mathcal{O}_A^\Gamma(X) \to \mathbb{M}$$

Wait, this is a Déjà Vu

$$\Gamma \vdash M : A \text{ and } x : X \in \Gamma \ \rightsquigarrow \ \mathbb{W}(\Gamma) \vdash [M]_x : \mathbb{C}(A)\langle [M] \rangle \to \mathfrak{M}\ \mathbb{C}(X)\langle x \rangle$$

A Bug In the Semi-Ring Matrix

What has been seen cannot be unseen

$$[x]_y\ \pi := \varnothing \qquad [x]_x\ \pi := \{\pi\}$$

$$[\lambda y.\ M]_x\ (y, \pi) := [M]_x\ \pi$$

$$[M\ N]_x\ \pi := \begin{array}{c} [M]_x\ ([N], \pi) \\ \oplus \\ ([M].2\ [N]\ \pi \gg\!= [N]_x) \end{array}$$

$$\frac{}{0\Gamma, x :_1 A \vdash x : A}$$

$$\frac{\Gamma, x :_\alpha A \vdash M : B}{\Gamma \vdash \lambda x.\ M : \Pi(x :_\alpha A).\ B}$$

$$\frac{\Gamma \vdash M : \Pi(x :_\alpha A).\ B \qquad \Delta \vdash N : A}{\Gamma + (\alpha \times \Delta) \vdash M\ N : B\{x := N\}}$$

Dialectica, Dialectica Everywhere

Dialectica is the HO dependent graded type of a term

- The semiring is replaced by its oidification (a monad + comm. add. monoid)
- Semiring values are now higher-order objects
- In particular they depend on the arguments

Dialectica, Dialectica Everywhere

Dialectica is the HO dependent graded type of a term

- The semiring is replaced by its oidification (a monad + comm. add. monoid)
- Semiring values are now higher-order objects
- In particular they depend on the arguments

This solves the graded expressivity issue

$\rightsquigarrow$ When pattern-matching over $M : \mathbb{B}$, $\alpha \sqcup \beta$ becomes basically

$$\texttt{if } M \texttt{ then } \alpha \texttt{ else } \beta$$

$\rightsquigarrow$ The argument of a function f has a now a highly dynamic annotation

$$\varphi : \Pi(x : \mathbb{W}(A))(\pi : \mathbb{C}(B)\langle f\ x\rangle).\, \mathfrak{M}\ \mathbb{C}(A)\langle x\rangle \qquad \text{(before: a fixed } \mathbb{M}\text{)}$$

So What?

I don't know what to do of this

- Is it known? Is it useful? It is practical?
- Can we get a decidable type system?
- Towards a full-blown synthetic complexity theory?

So What?

I don't know what to do of this

- Is it known? Is it useful? It is practical?
- Can we get a decidable type system?
- Towards a full-blown synthetic complexity theory?

What do you think about all this?

$\rightsquigarrow$ this is a call for help!

Thanks for your attention.