Constructing Unprejudiced Extensional Type Theories with Choices via Modalities

Liron Cohen and Vincent Rahli

August, 2022

Motivation

Time progressing elements are pervasive

- mutable references in programming languages
- choice sequence/forcing condition-like entities:
  - Forcing
  - Anti-classical theories: van Dalen (CT); Heyting (real analysis), Bridges & Richman (LEM), Kripke (MP); Coquand & Mannaa (MP); etc.

Motivation

Time progressing elements are pervasive

- mutable references in programming languages
- choice sequence/forcing condition-like entities:
  - Forcing
  - Anti-classical theories: van Dalen (CT); Heyting (real analysis), Bridges & Richman (LEM), Kripke (MP); Coquand & Mannaa (MP); etc.

Are choice sequences anti-classical?

Motivation

Time progressing elements are pervasive

- mutable references in programming languages
- choice sequence/forcing condition-like entities:
  - Forcing
  - Anti-classical theories: van Dalen (CT); Heyting (real analysis), Bridges & Richman (LEM), Kripke (MP); Coquand & Mannaa (MP); etc.

Are choice sequences anti-classical?

"Open Bar — a Brouwerian Intuitionistic Logic with a Pinch of Excluded Middle" (CSL'21): a classically inclined type theory with choice sequences

Motivation

Time progressing elements are pervasive

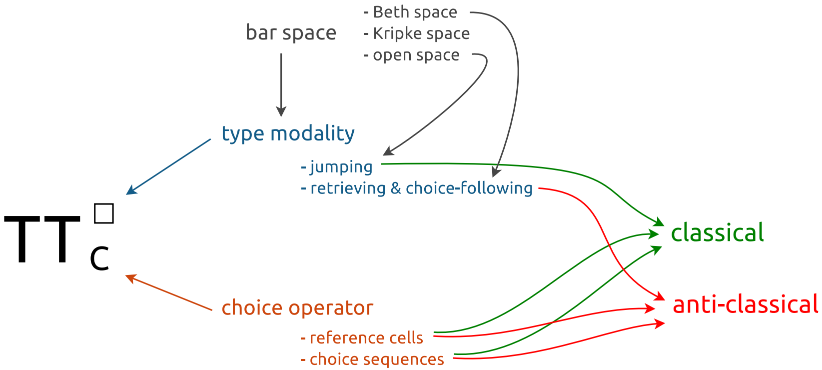
- mutable references in programming languages
- choice sequence/forcing condition-like entities:
  - Forcing
  - Anti-classical theories: van Dalen (CT); Heyting (real analysis), Bridges & Richman (LEM), Kripke (MP); Coquand & Mannaa (MP); etc.

Are choice sequences anti-classical?

"Open Bar — a Brouwerian Intuitionistic Logic with a Pinch of Excluded Middle" (CSL'21): a classically inclined type theory with choice sequences

Are choice sequences necessary?

This talk in 1 slide



Draws inspiration from

BITT

An anti-classical Brouwerian Intuitionsitic type theory with computable choice sequences and choice sequence axioms validated by a Beth model

Formalized in Coq

OpenTT

A classically-compatible Brouwerian Intuitionistic type theory with computable choice sequences and choice sequence axioms validated by a Beth-like *"open"* model

Formalized in Coq/Agda

BITT & OpenTT: Extensional Type Theories

Untyped call-by-name lambda-calculus

sequent calculus

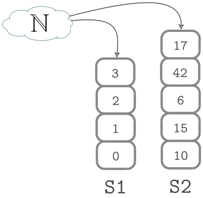
realizability semantics

Extensional

Dependent types

BITT & OpenTT: with Choice Sequences

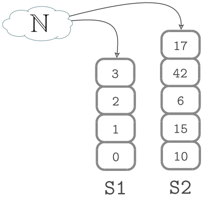
ETT $+$



Choice Sequences

BITT & OpenTT: with Choice Sequences

ETT +



Choice Sequences

Broader sense of computation

BITT & OpenTT: with Choice Sequences

lawless (free choice) sequences: no restrictions on the choices (except for initial segments)

LS$_1$ (density) $\forall s.\exists \alpha.\alpha \in s$

LS$_2$ (discreteness) $\forall \alpha, \beta.(\alpha \equiv \beta \ \vee \ \neg \alpha \equiv \beta)$

LS$_3$ (open data) $A(\alpha) \ \Rightarrow \ \exists n.\forall \beta.(\overline{\alpha} n = \overline{\beta} n \ \Rightarrow \ A(\beta))$

s	finite sequence
α	lawless sequence
$\alpha \in s$	s is an initial segment of α
$\equiv$	intensional equality
$\overline{\alpha} n$	the initial segment of α of length n

BITT & OpenTT: with Choice Sequences

BITT

LS1 $\Pi n{:}\mathbb{N}.\Pi f{:}\mathcal{B}_n.\Sigma\alpha{:}\texttt{Free}.f = \alpha \in \mathcal{B}_n$

LS2 $\Pi\alpha,\beta{:}\texttt{Free}.(\alpha{=}\beta \in \mathcal{B}){+}(\neg\alpha{=}\beta \in \mathcal{B})$

LS3 –

¬LEM $\neg\Pi P{:}\mathbb{P}.{\downarrow}(P{+}\neg P)$

¬MP $\neg\Pi P{:}\mathbb{B}^{\mathbb{N}}.\neg(\Pi n{:}\mathbb{N}.\neg P(n)) \rightarrow \Sigma n{:}\mathbb{N}.P(n)$

¬IP $\neg\Pi A{:}\mathbb{P}.\Pi B{:}\mathbb{P}^{\mathbb{N}}.(A \rightarrow \Sigma n{:}\mathbb{N}.B(n)) \rightarrow \Sigma n{:}\mathbb{N}.(A \rightarrow B(n))$

¬LPO $\neg\Pi P{:}\mathbb{B}^{\mathbb{N}}.(\Sigma\mathbb{N}{:}n.P(n)){+}(\Pi n{:}\mathbb{N}.\neg P(n))$

OpenTT

LS1 $\Pi n{:}\mathbb{N}.\Pi f{:}\mathcal{B}_n.{\downarrow}\Sigma\alpha{:}\texttt{Free}.f = \alpha \in \mathcal{B}_n$

LS2 $\Pi\alpha,\beta{:}\texttt{Free}.(\alpha{=}\beta \in \mathcal{B}){+}(\neg\alpha{=}\beta \in \mathcal{B})$

LS3 $\Pi\alpha{:}\texttt{Free}.P(\alpha) \rightarrow \Sigma n{:}\mathbb{N}_{\downarrow}.\Pi\beta{:}\texttt{Free}.(\alpha = \beta \in \mathcal{B}_{\downarrow n} \rightarrow {\downarrow}P(\beta))$

LEM $\Pi P{:}\mathbb{P}.{\downarrow}(P{+}\neg P)$

(where $\mathcal{B} = \mathbb{N}^{\mathbb{N}}$ and $\mathcal{B}_n = \mathbb{N}^{\mathbb{N}_n}$)
(↓ is a "proof erasure" operator)

BITT & OpenTT: Syntax

Syntax:

$$
\begin{array}{lll}
T \in \texttt{Type} & ::= & \mathbb{N} \mid \mathbb{U}_i \mid \mathbf{\Pi} x{:}t.t \mid \mathbf{\Sigma} x{:}t.t \mid \{x : t \mid t\} \\
 & \mid & t = t \in t \mid t{+}t \mid \ldots \\
 & \mid & \texttt{Free} \text{ (choice sequence type)}
\end{array}
$$

$$
\begin{array}{lll}
v \in \texttt{Value} & ::= & T \mid \star \mid \underline{n} \mid \lambda x.t \mid \langle t, t \rangle \mid \texttt{inl}(t) \mid \texttt{inr}(t) \mid \ldots \\
 & \mid & \eta \text{ (choice sequence name)}
\end{array}
$$

$$
\begin{array}{lll}
t \in \texttt{Term} & ::= & x \mid v \mid t\ t \mid \texttt{fix}(t) \mid \texttt{let}\ x := t\ \texttt{in}\ t \\
 & \mid & \texttt{case}\ t\ \texttt{of}\ \texttt{inl}(x) \Rightarrow t\ \mathbf{|}\ \texttt{inr}(y) \Rightarrow t \\
 & \mid & \texttt{let}\ x, y = t\ \texttt{in}\ t \mid \texttt{if}\ t{=}t\ \texttt{then}\ t\ \texttt{else}\ t \mid \ldots
\end{array}
$$

BITT & OpenTT: World-Based Computations

Operational semantics:

$$
\begin{array}{lcll}
w & \vdash & (\lambda x.t_1)\ t_2 & \longmapsto & t_1[x\backslash t_2] \\
w & \vdash & \texttt{let}\ x_1, x_2 = \langle t_1, t_2 \rangle\ \texttt{in}\ t & \longmapsto & t[x_1\backslash t_1; x_2\backslash t_2] \\
w & \vdash & \texttt{fix}(v) & \longmapsto & v\ \texttt{fix}(v)
\end{array}
$$

$\dots$

where $w \in \mathcal{W}$ (a poset)

BITT & OpenTT: World-Based Computations

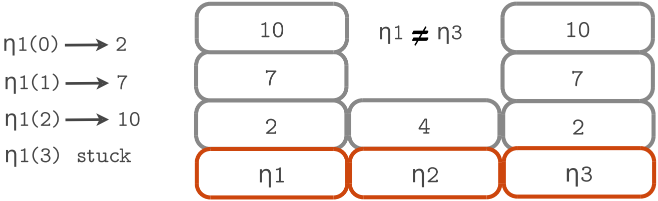
Operational semantics:

$$
\begin{array}{lcll}
w & \vdash & (\lambda x.t_1)\ t_2 & \longmapsto & t_1[x\backslash t_2] \\
w & \vdash & \texttt{let}\ x_1, x_2 = \langle t_1, t_2 \rangle\ \texttt{in}\ t & \longmapsto & t[x_1\backslash t_1; x_2\backslash t_2] \\
w & \vdash & \texttt{fix}(v) & \longmapsto & v\ \texttt{fix}(v)
\end{array}
$$

...

where $w \in \mathcal{W}$ (a poset)

World-dependent operational semantics:



BITT & OpenTT: Inference Rule

Standard ETT rules:

$$\frac{\Gamma, x : A \vdash b : B[x] \qquad \Gamma \vdash \star : (A \in \mathbb{U}_i)}{\Gamma \vdash \lambda x.b : \mathbf{\Pi} a{:}A.B[a]} \qquad \cdots$$

BITT & OpenTT: Inference Rule

Standard ETT rules:

$$\frac{\Gamma, x : A \vdash b : B[x] \qquad \Gamma \vdash \star : (A \in \mathbb{U}_i)}{\Gamma \vdash \lambda x.b : \mathbf{\Pi} a{:}A.B[a]} \qquad \cdots$$

+ choice sequence rules:

$$\frac{}{\Gamma \vdash \star : (\eta \in \mathtt{Free})} \qquad \frac{}{\Gamma \vdash \star : (\eta \in \mathcal{B})} \qquad \cdots$$

BITT & OpenTT: Inference Rule

Standard ETT rules:

$$\frac{\Gamma, x : A \vdash b : B[x] \qquad \Gamma \vdash \star : (A \in \mathbb{U}_i)}{\Gamma \vdash \lambda x.b : \mathbf{\Pi} a{:}A.B[a]} \qquad \cdots$$

+ choice sequence rules:

$$\frac{}{\Gamma \vdash \star : (\eta \in \texttt{Free})} \qquad \frac{}{\Gamma \vdash \star : (\eta \in \mathcal{B})} \qquad \cdots$$

+ LS1 (density), LS2 (discreteness), LS3 (Open Data)

BITT & OpenTT: Inference Rule

Standard ETT rules:

$$\frac{\Gamma, x : A \vdash b : B[x] \qquad \Gamma \vdash \star : (A \in \mathbb{U}_i)}{\Gamma \vdash \lambda x.b : \mathbf{\Pi} a{:}A.B[a]} \qquad \cdots$$

+ choice sequence rules:

$$\frac{}{\Gamma \vdash \star : (\eta \in \mathtt{Free})} \qquad \frac{}{\Gamma \vdash \star : (\eta \in \mathcal{B})} \qquad \cdots$$

+ LS1 (density), LS2 (discreteness), LS3 (Open Data)

+ LEM in OpenTT & $\neg$LEM in BITT

BITT & OpenTT: Realizability semantics

An inductive relation that expresses type equality

$$w \vDash T_1 \equiv T_2$$

A recursive function that expresses equality in a type

$$w \vDash a \equiv b \in T$$

BITT & OpenTT: Realizability semantics

An inductive relation that expresses type equality

$$w \vDash T_1 \equiv T_2$$

A recursive function that expresses equality in a type

$$w \vDash a \equiv b \in T$$

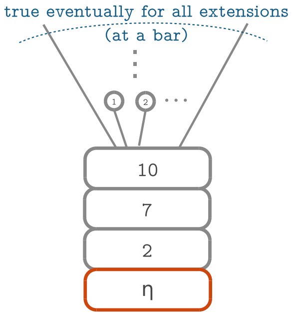
For example (product types):

$$w \vDash \mathbf{\Pi} x_1{:}A_1.B_1 \equiv \mathbf{\Pi} x_2{:}A_2.B_2$$

$$\forall w' \sqsupseteq w. w' \vDash A_1 \equiv A_2 \land$$
$$\forall w' \sqsupseteq w. \forall a_1, a_2.\ w' \vDash a_1 \equiv a_2 \in A_1 \Rightarrow w' \vDash B_1[x_1 \backslash a_1] \equiv B_2[x_2 \backslash a_2]$$

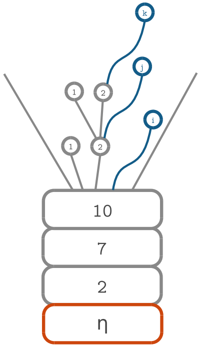
BITT vs. OpenTT

Beth model



$w \vDash T \iff$
$\exists b \in \mathrm{bar}(w).\forall w_1 \in b.$
$\forall w_2 \sqsupseteq w_1.w_2 \vDash T$
(a modality: $\Box_B(T)$)

Open Bar model



$w \vDash T \iff$
$\forall w_1 \sqsupseteq w.\exists w_2 \sqsupseteq w_1.$
$\forall w_3 \sqsupseteq w_2.w_2 \vDash T$
(a modality: $\Box_O(T)$)

From BITT & OpenTT to $\mathsf{TT}_{\mathcal{C}}^{\square}$

This can all be generalized as follows...

Formalized in Agda

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Choice Operator

Components

- $\mathcal{N}$: abstract type of choice names
- $\mathcal{C}$: abstract type of choices inhabited by two distinct choices $\boldsymbol{\kappa_0}$ and $\boldsymbol{\kappa_1}$
- a partial function: $\mathsf{choice?} \in \mathcal{W} \to \mathcal{N} \to \mathbb{N} \to \mathcal{C}$

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: Choice Operator

Components

- $\mathcal{N}$: abstract type of choice names
- $\mathcal{C}$: abstract type of choices inhabited by two distinct choices $\boldsymbol{\kappa_0}$ and $\boldsymbol{\kappa_1}$
- a partial function: $\mathsf{choice?} \in \mathcal{W} \to \mathcal{N} \to \mathbb{N} \to \mathcal{C}$

Syntax

$$v \in \mathtt{Value} ::= \cdots \mid \delta \text{ (choice name)}$$

$\mathrm{TT}_{\mathcal{C}}^{\Box}$: Choice Operator

Components

- $\mathcal{N}$: abstract type of choice names
- $\mathcal{C}$: abstract type of choices inhabited by two distinct choices $\boldsymbol{\kappa_0}$ and $\boldsymbol{\kappa_1}$
- a partial function: $\text{choice?} \in \mathcal{W} \to \mathcal{N} \to \mathbb{N} \to \mathcal{C}$

Syntax

$$v \in \texttt{Value} ::= \cdots \mid \delta \text{ (choice name)}$$

Operational Semantics

$$w \vdash \delta(\underline{n}) \mapsto \text{choice?}(w, \delta, n)$$

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Modality

An abstract modality on (the semantics of) types: $\square$

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Modality

An abstract modality on (the semantics of) types: $\square$

Forcing interpretation: $\square_w(w'.w' \vDash T) \to w \vDash T$

$\text{TT}_{\mathcal{C}}^{\Box}$: Modality

An abstract modality on (the semantics of) types: $\Box$

Forcing interpretation: $\Box_w(w'.w' \vDash T) \rightarrow w \vDash T$

Properties:

monotonicity of $\Box$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\forall w' \sqsupseteq w.\Box_w P \rightarrow \Box_{w'} P$
K, distribution axiom	$\forall(w:\mathcal{W})(P,Q:\mathcal{P}_w).\Box_w(P \rightarrow Q) \rightarrow \Box_w P \rightarrow \Box_w Q$
$C4$, i.e., $\Box\Box \rightarrow \Box$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\Box_w(w'.\Box_{w'} P) \rightarrow \Box_w P$
$\forall \rightarrow \Box$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\forall_w^{\sqsubseteq}(P) \rightarrow \Box_w P$
T, reflexivity axiom	$\forall(w:\mathcal{W})(P:\mathbb{P}).\Box_w(w'.P) \rightarrow P$

$\mathrm{TT}_{\mathcal{C}}^{\square}$: Modality

An abstract modality on (the semantics of) types: $\square$

Forcing interpretation: $\square_w(w'.w' \vDash T) \rightarrow w \vDash T$

Properties:

monotonicity of $\square$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\forall w' \sqsupseteq w.\square_w P \rightarrow \square_{w'} P$
K, distribution axiom	$\forall(w:\mathcal{W})(P,Q:\mathcal{P}_w).\square_w(P \rightarrow Q) \rightarrow \square_w P \rightarrow \square_w Q$
$C4$, i.e., $\square\square \rightarrow \square$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\square_w(w'.\square_{w'} P) \rightarrow \square_w P$
$\forall \rightarrow \square$	$\forall(w:\mathcal{W})(P:\mathcal{P}_w).\forall_w^{\sqsubseteq}(P) \rightarrow \square_w P$
T, reflexivity axiom	$\forall(w:\mathcal{W})(P:\mathbb{P}).\square_w(w'.P) \rightarrow P$

Enough to prove the standard properties of the type system: consistency, symmetry, transitivity, etc.

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Bar Space

Modalities can be derived from *bar spaces*

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: Bar Space

Modalities can be derived from *bar spaces*

Opens: $\mathcal{O} \coloneqq \mathcal{W} \to \mathbb{P}$ (predicates on worlds)

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Bar Space

Modalities can be derived from *bar spaces*

Opens: $\mathcal{O} \coloneqq \mathcal{W} \rightarrow \mathbb{P}$ (predicates on worlds)

Predicates on opens: $\mathsf{BarProp} \coloneqq \mathcal{W} \rightarrow \mathcal{O} \rightarrow \mathbb{P}$

$\mathrm{TT}_{\mathcal{C}}^{\square}$: Bar Space

Modalities can be derived from *bar spaces*

Opens: $\mathcal{O} \coloneqq \mathcal{W} \to \mathbb{P}$ (predicates on worlds)

Predicates on opens: $\mathsf{BarProp} \coloneqq \mathcal{W} \to \mathcal{O} \to \mathbb{P}$

$B \in \mathsf{BarProp}$ is a *bar space* if:

- it is closed under binary intersections, union & subsets
- it contains the top element
- its elements are non-empty

$\mathsf{TT}^{\square}_{\mathcal{C}}$: Bar Space

Modalities can be derived from *bar spaces*

Opens: $\mathcal{O} \coloneqq \mathcal{W} \to \mathbb{P}$ (predicates on worlds)

Predicates on opens: $\mathsf{BarProp} \coloneqq \mathcal{W} \to \mathcal{O} \to \mathbb{P}$

$B \in \mathsf{BarProp}$ is a *bar space* if:

- it is closed under binary intersections, union & subsets
- it contains the top element
- its elements are non-empty

Any bar space $B \in \mathsf{BarProp}$ can be turned into a modality $\square$

$TT_{\mathcal{C}}^{\square}$: Bar Space – Examples

Kripke bar space

$$\mathsf{Kripke} \coloneqq \lambda w.\lambda o.\forall_{w}^{\sqsubseteq}(w'.w' \in o)$$

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: Bar Space – Examples

Kripke bar space

$$\mathsf{Kripke} \coloneqq \lambda w.\lambda o.\forall_{w}^{\sqsubseteq}(w'.w' \in o)$$

Beth bar space

$$\mathsf{Beth} \coloneqq \lambda w.\lambda o.\forall(c : \mathsf{chain}(w)).\mathsf{barred}(o, c)$$

$\mathsf{TT}_{\mathcal{C}}^{\square}$: Bar Space – Examples

Kripke bar space

$$\mathsf{Kripke} \coloneqq \lambda w.\lambda o.\forall_{w}^{\sqsubseteq}(w'.w' \in o)$$

Beth bar space

$$\mathsf{Beth} \coloneqq \lambda w.\lambda o.\forall(c : \mathsf{chain}(w)).\mathsf{barred}(o, c)$$

open bar space

$$\mathsf{Open} \coloneqq \lambda w.\lambda o.\forall_{w}^{\sqsubseteq}(w_1.\exists_{w_1}^{\sqsubseteq}(w_2.w_2 \in o))$$

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: LEM and ¬LEM – ¬LEM requirements

We require the following components:

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: LEM and $\neg$LEM – $\neg$LEM requirements

We require the following components:

- ability to create new choice names and new choices

$TT_{\mathcal{C}}^{\square}$: LEM and $\neg$LEM – $\neg$LEM requirements

We require the following components:

- ability to create new choice names and new choices
- ability to make an immutable choice

$\mathrm{TT}_{\mathcal{C}}^{\square}$: LEM and ¬LEM – ¬LEM requirements

We require the following components:

- ability to create new choice names and new choices
- ability to make an immutable choice
- an object-level type of choices $\mathrm{Type}\mathcal{C}$

$\text{TT}_{\mathcal{C}}^{\square}$: LEM and ¬LEM – ¬LEM requirements

We require the following components:

- ability to create new choice names and new choices
- ability to make an immutable choice
- an object-level type of choices $\text{Type}\mathcal{C}$
- that $\square$ is *retrieving*: $\square_w(w'.\text{choice?}(w', \delta, n)$ is defined$)$

$\mathrm{TT}_{\mathcal{C}}^{\Box}$: LEM and ¬LEM – ¬LEM requirements

We require the following components:

- ability to create new choice names and new choices
- ability to make an immutable choice
- an object-level type of choices $\mathsf{Type}\mathcal{C}$
- that $\Box$ is *retrieving*: $\Box_w(w'.\mathsf{choice?}(w', \delta, n)$ is defined)
- that $\Box$ is *choice-following*:
  - if δ's only choice is κ in w
  - and $\Box_w P$
  - then there exists $w' \sqsupseteq w$ such that $P(w')$ and δ's only choice is κ in w'

$\mathsf{TT}_{\mathcal{C}}^{\square}$: LEM and $\neg$LEM – Crux of $\neg$LEM's proof

$\text{TT}_{\mathcal{C}}^{\square}$: LEM and ¬LEM – Crux of ¬LEM's proof

- assume $w \vDash \mathbf{\Pi} P{:}\mathbb{P}.\downarrow(P+\neg P)$, prove $\bot$
- w: a world & δ: a new choice w.r.t. w
- $\Sigma\mathcal{C} := \mathbf{\Sigma} k{:}\mathbb{N}.(\delta(k))=\boldsymbol{\kappa_1} \in \text{Type}\mathcal{C}$ (an equality type)

$\mathrm{TT}_{\mathcal{C}}^{\Box}$: LEM and ¬LEM – Crux of ¬LEM's proof

- assume $w \vDash \mathbf{\Pi} P{:}\mathbb{P}.{\downarrow}(P{+}\neg P)$, prove $\perp$
- w: a world & δ: a new choice w.r.t. w
- $\Sigma\mathcal{C} := \mathbf{\Sigma} k{:}\mathbb{N}.(\delta(k)){=}\boldsymbol{\kappa_1} \in \mathrm{Type}\mathcal{C}$ (an equality type)

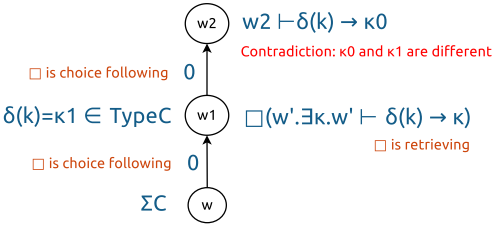
Only δ-choice so far in w is $\boldsymbol{\kappa_0}$

$TT^{\square}_{\mathcal{C}}$: LEM and ¬LEM – Crux of ¬LEM's proof

- assume $w \vDash \mathbf{\Pi} P{:}\mathbb{P}.{\downarrow}(P + \neg P)$, prove $\perp$
- w: a world & δ: a new choice w.r.t. w
- $\Sigma\mathcal{C} := \mathbf{\Sigma} k{:}\mathbb{N}.(\delta(k)) = \boldsymbol{\kappa_1} \in \mathsf{Type}\mathcal{C}$ (an equality type)

Only δ-choice so far in w is $\boldsymbol{\kappa_0}$

- $\neg w \vDash \Sigma\mathcal{C}$: Assume $\Sigma\mathcal{C}$.

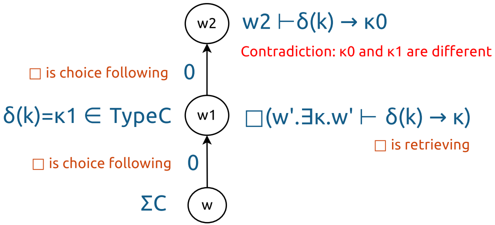


$TT^{\square}_{\mathcal{C}}$: LEM and $\neg$LEM – Crux of $\neg$LEM's proof

- assume $w \vDash \mathbf{\Pi} P{:}\mathbb{P}.{\downarrow}(P + \neg P)$, prove $\perp$
- w: a world & δ: a new choice w.r.t. w
- $\Sigma\mathcal{C} := \mathbf{\Sigma} k{:}\mathbb{N}.(\delta(k)) {=} \boldsymbol{\kappa_1} \in \text{Type}\mathcal{C}$ (an equality type)

Only δ-choice so far in w is $\boldsymbol{\kappa_0}$

- $\neg w \vDash \Sigma\mathcal{C}$: Assume $\Sigma\mathcal{C}$.



- $\neg\forall^{\sqsubseteq}_{w'}(w''.\neg w'' \vDash \Sigma\mathcal{C})$: Instantiate $\forall^{\sqsubseteq}_{w'}(w''.\neg w'' \vDash \Sigma\mathcal{C})$ with a world w_1 where $\boldsymbol{\kappa_1}$ is *immutable*. We get $\neg w_1 \vDash \Sigma\mathcal{C}$.

  Contradiction: we can also prove $w_1 \vDash \Sigma\mathcal{C}$

$\mathsf{TT}_{\mathcal{C}}^{\square}$: LEM and $\neg$LEM – LEM requirements

$\mathsf{TT}_{\mathcal{C}}^{\Box}$: LEM and $\neg$LEM – LEM requirements

$\Box$ is *jumping*:

$$\forall (w : \mathcal{W})(P : \mathcal{P}_w).\forall_w^{\sqsubseteq}(w_1.\exists_{w_1}^{\sqsubseteq}(w_2.\Box_{w_2} P)) \rightarrow \Box_w P$$

$TT_{\mathcal{C}}^{\square}$: Summary

$\square$	$\mathcal{C}$	classical?
Beth	*CS*	✗
Beth	*Ref*	✗
open	*CS*	✓
open	*Ref*	✓
Kripke	*CS*	?
Kripke	*Ref*	✗

$TT_{\mathcal{C}}^{\square}$: Summary

$\square$	$\mathcal{C}$	classical?
Beth	*CS*	✗
Beth	*Ref*	✗
open	*CS*	✓
open	*Ref*	✓
Kripke	*CS*	?
Kripke	*Ref*	✗

Beth is not *jumping*

Open is not *choice-following*

Kripke is not *retrieving* (with CS) and not *jumping*

$TT_{\mathcal{C}}^{\square}$: Summary

$\square$	$\mathcal{C}$	classical?
Beth	*CS*	✗
Beth	*Ref*	✗
open	*CS*	✓
open	*Ref*	✓
Kripke	*CS*	?
Kripke	*Ref*	✗

Beth is not *jumping*

Open is not *choice-following*

Kripke is not *retrieving* (with CS) and not *jumping*

Choice sequences are **not necessary** (references are enough)
Choice sequences (or references) are **not anti-classical**

$TT_{\mathcal{C}}^{\Box}$: Summary

$\Box$	$\mathcal{C}$	classical?
Beth	*CS*	✗
Beth	*Ref*	✗
open	*CS*	✓
open	*Ref*	✓
Kripke	*CS*	?
Kripke	*Ref*	✗

Beth is not *jumping*

Open is not *choice-following*

Kripke is not *retrieving* (with CS) and not *jumping*

Choice sequences are **not necessary** (references are enough)
Choice sequences (or references) are **not anti-classical**

Questions?