Derivability of (novel) rules of β-conversion in partial type theory

Petr Kuchyňka & Jiří Raclavský

Dpt. of Philosophy, Masaryk University (Brno)

MUNI

Czech Gathering of Logicians, 16/6/2022, Prague

Motivation example

- High-school mathematics: find the domain $\mathscr{D}_f$ of the function prescribed by

$$y = \frac{3}{x^2 - 3x + 2}$$

- $f := \lambda x.\frac{3}{(x-1)\times(x-2)}$,

  so $\mathscr{D}_f = \mathbb{R} - \{1;2\}$

- Verification by application of f to 1 (and 2), i.e. by β-reduction:

  both $f(1)$ and $f(2)$ lead to *nothing at all*

Motivation example

- High-school mathematics: find the domain $\mathscr{D}_f$ of the function prescribed by

$$y = \frac{3}{x^2 - 3x + 2}$$

- $f := \lambda x. \frac{3}{(x-1)\times(x-2)}$,

  so $\mathscr{D}_f = \mathbb{R} - \{1; 2\}$

- Verification by application of f to 1 (and 2), i.e. by β-reduction:

  both $f(1)$ and $f(2)$ lead to *nothing at all*

Motivation example

- High-school mathematics: find the domain $\mathscr{D}_f$ of the function prescribed by

$$y = \frac{3}{x^2 - 3x + 2}$$

- $f := \lambda x.\frac{3}{(x-1)\times(x-2)}$,

  so $\quad \mathscr{D}_f = \mathbb{R} - \{1; 2\}$

- Verification by application of f to 1 (and 2), i.e. by β-reduction: both $f(1)$ and $f(2)$ lead to *nothing at all*

Motivation example

- High-school mathematics: find the domain $\mathscr{D}_f$ of the function prescribed by

$$y = \frac{3}{x^2 - 3x + 2}$$

- $f := \lambda x.\frac{3}{(x-1)\times(x-2)}$,

  so $\mathscr{D}_f = \mathbb{R} - \{1; 2\}$

- Verification by application of f to 1 (and 2), i.e. by β-reduction:

  both $f(1)$ and $f(2)$ lead to *nothing at all*

Abstract

- The β*-reduction rule* $[\lambda x.f(x)]\ (a) \vdash f(x)_{(a/x)}$

  is the *fundamental* rule of λ-calculus/**type theory** (**TT**)
- TT is the(?) *higher-order logic* (*HOL*); vast applications, e.g. in
  a. functional programming languages
  b. proof assistants
  c. formalisation of natural language
- **Partial TT** meets the widespread demand of *CS* to employ *partiality*

1. show *appropriate rules* of β-conversion by-name and by-value
2. formulate *novel rules* β-conversion by-name and by-value
3. *derive* all the rules from the initial couple of β-conversion by-name rules

Abstract

- The *β-reduction rule* $[\lambda x.f(x)]\ (a) \vdash f(x)_{(a/x)}$

  is the *fundamental* rule of λ-calculus/**type theory** (**TT**)

- TT is the(?) *higher-order logic* (*HOL*); vast applications, e.g. in
  a. functional programming languages
  b. proof assistants
  c. formalisation of natural language

- **Partial TT** meets the widespread demand of *CS* to employ *partiality*

1. show *appropriate rules* of β-conversion by-name and by-value
2. formulate *novel rules* β-conversion by-name and by-value
3. *derive* all the rules from the initial couple of β-conversion by-name rules

Abstract

- The *β-reduction rule* $[\lambda x.f(x)]\ (a) \vdash f(x)_{(a/x)}$

  is the *fundamental* rule of λ-calculus/**type theory** (**TT**)

- TT is the(?) *higher-order logic* (*HOL*); vast applications, e.g. in

  a. functional programming languages
  b. proof assistants
  c. formalisation of natural language

- **Partial TT** meets the widespread demand of *CS* to employ *partiality*

1. show *appropriate rules* of β-conversion by-name and by-value
2. formulate *novel rules* β-conversion by-name and by-value
3. *derive* all the rules from the initial couple of β-conversion by-name rules

Abstract

- The *β-reduction rule* $[\lambda x.f(x)]\,(a) \vdash f(x)_{(a/x)}$

  is the *fundamental* rule of λ-calculus/**type theory** (**TT**)

- TT is the(?) *higher-order logic* (*HOL*); vast applications, e.g. in
  a. functional programming languages
  b. proof assistants
  c. formalisation of natural language

- **Partial TT** meets the widespread demand of *CS* to employ *partiality*

1. show *appropriate rules* of β-conversion by-name and by-value
2. formulate *novel rules* β-conversion by-name and by-value
3. *derive* all the rules from the initial couple of β-conversion by-name rules

Content

Non-denoting terms and partiality

Non-denoting terms

Non-denoting terms are terms that lack denotation.

- e.g., "$3 \div 0$", "$3 - 10$" (in $\mathbb{N}$), "*the greatest prime*", "*the King of France*", ...

- *Logics* managing non-denoting terms:
  a. *partial type theory* – Tichý 1982 and *passim*, Farmer 1990, Muskens 1989, 1995, Duží et al. 2010, Raclavský et al. 2015, ...,
  b. *(typed) lambda calculus* – Besson 1984, Moggi 1988, Feferman 1995, Moschovakis 2006, ...
  c. *free logic* – Lambert 2003, Scott 1979, ...
  d. *three-(four-)valued logic* – Bochvar 1981, Kleene 1952, Blamey 1986, Langholm 1988, Muskens 1995, Kohlhase 1996, ...
  e. *fuzzy logic* – Novák + B. 2015, Běhounek 2015, B. + Daňková 2016, B. + Dvořák 2018, ...

Non-denoting terms and partiality

Non-denoting terms

Non-denoting terms are terms that lack denotation.

– e.g., "$3 \div 0$", "$3 - 10$" (in $\mathbb{N}$), "*the greatest prime*", "*the King of France*", ...

- *Logics* managing non-denoting terms:
  a. *partial type theory* – Tichý 1982 and *passim*, Farmer 1990, Muskens 1989, 1995, Duží et al. 2010, Raclavský et al. 2015, ...,
  b. *(typed) lambda calculus* – Besson 1984, Moggi 1988, Feferman 1995, Moschovakis 2006, ...
  c. *free logic* – Lambert 2003, Scott 1979, ...
  d. *three-(four-)valued logic* – Bochvar 1981, Kleene 1952, Blamey 1986, Langholm 1988, Muskens 1995, Kohlhase 1996, ...
  e. *fuzzy logic* – Novák + B. 2015, Běhounek 2015, B. + Daňková 2016, B. + Dvořák 2018, ...

Two kinds of partiality

Two kinds of partiality

'Functional terms' such as "$\lambda x. \div (x, 0)$"

a. *denote* **partial functions-as-graphs**

b. *express* **strict** ('partial') **functions-as-computations**

- b. is of a profound interest in *computer science*
  - Besson, Moggi, Feferman, ...

Partial functions (a.-type partiality)

Partial functions

A **partial function[-as-graph]** is *undefined* for at least 1 element of its domain $\mathscr{D}_F$.

i.e. not all members of its $\mathscr{D}_F$ are mapped to its co-domain $\mathscr{C}_F$.

– *Example.* Let $x, y/\tau^{\mathbb{R}}$ (*type* of reals); $\div/\langle\tau^{\mathbb{R}}, \tau^{\mathbb{R}}\rangle \mapsto \tau^{\mathbb{R}}$;

$\langle x, y\rangle$	$\div(x, y)$
$\vdots$	$\vdots$
$\langle 3, 0\rangle$	i.e. nothing at all
$\langle 3, 1\rangle$	3
$\langle 3, 2\rangle$	1.5
$\vdots$	$\vdots$

Partial functions (a.-type partiality)

Partial functions

A **partial function[-as-graph]** is *undefined* for at least 1 element of its domain $\mathscr{D}_F$.

i.e. not all members of its $\mathscr{D}_F$ are mapped to its co-domain $\mathscr{C}_F$.

– *Example.* Let $x, y/\tau^{\mathbb{R}}$ (*type* of reals); $\div/\langle\tau^{\mathbb{R}}, \tau^{\mathbb{R}}\rangle \mapsto \tau^{\mathbb{R}}$;

$\langle x, y\rangle$	$\div(x, y)$
$\vdots$	$\vdots$
$\langle 3, 0\rangle$	i.e. nothing at all
$\langle 3, 1\rangle$	3
$\langle 3, 2\rangle$	1.5
$\vdots$	$\vdots$

Constructions (b.-type partiality)

Constructions

A **construction** C (Tichý, Girard, Moschovakis, ...) is an acyclic algorithmic computation of an object O *denoted* by the term "C" that *expresses* C.

E.g., "$3{\div}0$" expresses the construction $\div(3,0)$ and denotes nothing at all.

Let v be an *assignment* for variables-as-constructions.

(Im)Proper constructions

C is v-**proper** / v-**improper** iff C v-constructs an object / nothing at all.

v-congruence of constructions

C and D are v-**congruent**, $\cong$ iff C and D v-construct the same object, or they are both v-improper. e.g., $\div(3,1)$ and $+(1,2)$, $\div(3,0)$ and $\div(2,0)$

Constructions (b.-type partiality)

Constructions

A **construction** C (Tichý, Girard, Moschovakis, ...) is an acyclic algorithmic computation of an object O *denoted* by the term "C" that *expresses* C.

E.g., "$3 \div 0$" expresses the construction $\div(3,0)$ and denotes nothing at all.

Let v be an *assignment* for variables-as-constructions.

(Im)Proper constructions

C is v-**proper** / v-**improper** iff C v-constructs an object / nothing at all.

v-congruence of constructions

C and D are v-**congruent**, $\cong$ iff C and D v-construct the same object, or they are both v-improper. e.g., $\div(3,1)$ and $+(1,2)$, $\div(3,0)$ and $\div(2,0)$

Language of TT*

Partial type theory (PTT)

Partial type theory (**PTT**) manages both total and partial functions.

(Tichý 1982, Farmer 1990)

Language $\mathcal{L}_{\mathsf{TT}^*}$

$$C ::= \quad x \quad | \quad \mathrm{c} \quad | \quad C_0(\bar{C}_m) \quad | \quad \lambda\tilde{x}_m.C_0 \quad | \quad \ulcorner C_0 \urcorner$$

- i.e. variables, constants, *applications*, *abstractions*, *acquisitions*
- where $\bar{X}_m$ stands for $X_1, ..., X_m$ and $\tilde{X}_m$ stands for $X_1...X_m$
- auxiliary brackets: [,]
- *typing*: C/τ where *type* τ is either ι or o of $\mathscr{B}$, or $\langle\bar{\tau}_m\rangle \rightarrow \tau_0$ over $\mathscr{B}$

- *Semantics* (based on Henkin 1950):
  - a *domain* $\mathscr{D}_\tau$ is a set that interprets τ
  - $\mathscr{F} = \{\mathscr{D}_\tau\}_\tau$ is a *frame* and $\mathscr{M} = \langle\mathscr{F}, \mathscr{I}\rangle$ a (general) *model*

Language of TT*

Partial type theory (PTT)

Partial type theory (**PTT**) manages both total and partial functions.

(Tichý 1982, Farmer 1990)

Language $\mathcal{L}_{\mathsf{TT}^*}$

$$C ::= \quad x \quad | \quad \mathrm{c} \quad | \quad C_0(\bar{C}_m) \quad | \quad \lambda\tilde{x}_m.C_0 \quad | \quad \ulcorner C_0 \urcorner$$

- i.e. variables, constants, *applications*, *abstractions*, *acquisitions*
- where $\bar{X}_m$ stands for $X_1, ..., X_m$ and $\tilde{X}_m$ stands for $X_1 ... X_m$
- auxiliary brackets: [,]
- *typing*: C/τ where *type* τ is either ι or o of $\mathscr{B}$, or $\langle\bar{\tau}_m\rangle{\rightarrow}\tau_0$ over $\mathscr{B}$

- *Semantics* (based on Henkin 1950):
  - a *domain* $\mathscr{D}_\tau$ is a set that interprets τ
  - $\mathscr{F} = \{\mathscr{D}_\tau\}_\tau$ is a *frame* and $\mathscr{M} = \langle\mathscr{F}, \mathscr{I}\rangle$ a (general) *model*

Language of TT*

Partial type theory (PTT)

Partial type theory (**PTT**) manages both total and partial functions.

(Tichý 1982, Farmer 1990)

Language $\mathcal{L}_{\mathrm{TT}^*}$

$$C ::= \quad x \quad | \quad \mathrm{c} \quad | \quad C_0(\bar{C}_m) \quad | \quad \lambda\tilde{x}_m.C_0 \quad | \quad \ulcorner C_0 \urcorner$$

- i.e. variables, constants, *applications*, *abstractions*, *acquisitions*
- where $\bar{X}_m$ stands for $X_1, ..., X_m$ and $\tilde{X}_m$ stands for $X_1 ... X_m$
- auxiliary brackets: [,]
- *typing*: C/τ where *type* τ is either ι or o of $\mathscr{B}$, or $\langle\bar{\tau}_m\rangle{\rightarrow}\tau_0$ over $\mathscr{B}$

- *Semantics* (based on Henkin 1950):
  - a *domain* $\mathscr{D}_\tau$ is a set that interprets τ
  - $\mathscr{F} = \{\mathscr{D}_\tau\}_\tau$ is a *frame* and $\mathscr{M} = \langle\mathscr{F}, \mathscr{I}\rangle$ a (general) *model*

Substitution

Substitution as a construction

$C_{(D/x)} = [\![Sub(\ulcorner D \urcorner, \ulcorner x \urcorner, \ulcorner C \urcorner)]\!]^{\mathscr{M},v}$

Substitution function Sub v-constructed by Sub

Let $C, D, x, B, \bar{B}_m$ are nth-order constructions. Let "$FV(C)$" stand for the set of all free variables that are (nth-order) subconstructions of C. (adaptated from Curry 1958)

I. If the variable x is *not free* (λ, $\ulcorner\urcorner$ binds) in C, then $C_{(D/x)}$ is identical with C.

II. If the variable x is *free* in C, then

	If C is ...	*$C_{(D/x)}$ is ...*	*condition:*
i.	x	D	
ii.	$B(\bar{B}_m)$	$B_{(D/x)}(\bar{B}_{m(D/x)})$	
iii.	$\lambda y.B$	$\lambda y.B_{(D/x)}$	$x \in FV(B)$ and $y \notin FV(D)$
iv.	$\lambda y.B$	$[\lambda z.B_{(z/y)}]_{(D/x)}$	$x \in FV(B)$ and $y \in FV(D)$ and $z \notin FV(B) \cup FV(D)$

- *Substitution principle*: For any v into $\mathscr{F}$ and any C, if $[\![D]\!]^{\mathscr{M},v} = \mathrm{D}$, then $[\![C_{(D/x)}]\!]^{\mathscr{M},v} = [\![C]\!]^{\mathscr{M},v(\mathrm{D}/x)}$.

Substitution

Substitution as a construction

$C_{(D/x)} = [\![Sub(\ulcorner D\urcorner, \ulcorner x\urcorner, \ulcorner C\urcorner)]\!]^{\mathscr{M},v}$

Substitution function Sub v-constructed by Sub

Let $C, D, x, B, \bar{B}_m$ are nth-order constructions. Let "$FV(C)$" stand for the set of all free variables that are (nth-order) subconstructions of C. (adaptated from Curry 1958)

I. If the variable x is *not free* (λ, $\ulcorner\urcorner$ binds) in C, then $C_{(D/x)}$ is identical with C.

II. If the variable x is *free* in C, then

	If C is ...	*$C_{(D/x)}$ is ...*	*condition:*
i.	x	D	
ii.	$B(\bar{B}_m)$	$B_{(D/x)}(\bar{B}_{m(D/x)})$	
iii.	$\lambda y.B$	$\lambda y.B_{(D/x)}$	$x \in FV(B)$ and $y \notin FV(D)$
iv.	$\lambda y.B$	$[\lambda z.B_{(z/y)}]_{(D/x)}$	$x \in FV(B)$ and $y \in FV(D)$ and $z \notin FV(B) \cup FV(D)$

- *Substitution principle*: For any v into $\mathscr{F}$ and any C, if $[\![D]\!]^{\mathscr{M},v} = \mathrm{D}$, then $[\![C_{(D/x)}]\!]^{\mathscr{M},v} = [\![C]\!]^{\mathscr{M},v(\mathrm{D}/x)}$.

Substitution

Substitution as a construction

$C_{(D/x)} = [\![Sub(\ulcorner D \urcorner, \ulcorner x \urcorner, \ulcorner C \urcorner)]\!]^{\mathscr{M},v}$

Substitution function Sub v-constructed by Sub

Let $C, D, x, B, \bar{B}_m$ are nth-order constructions. Let "$FV(C)$" stand for the set of all free variables that are (nth-order) subconstructions of C. (adaptated from Curry 1958)

I. If the variable x is *not free* (λ, $\ulcorner\urcorner$ binds) in C, then $C_{(D/x)}$ is identical with C.

II. If the variable x is *free* in C, then

	If C is ...	*$C_{(D/x)}$ is ...*	*condition:*
i.	x	D	
ii.	$B(\bar{B}_m)$	$B_{(D/x)}(\bar{B}_{m(D/x)})$	
iii.	$\lambda y.B$	$\lambda y.B_{(D/x)}$	$x \in FV(B)$ and $y \notin FV(D)$
iv.	$\lambda y.B$	$[\lambda z.B_{(z/y)}]_{(D/x)}$	$x \in FV(B)$ and $y \in FV(D)$ and $z \notin FV(B) \cup FV(D)$

- *Substitution principle*: For any v into $\mathscr{F}$ and any C, if $[\![D]\!]^{\mathscr{M},v} = \mathrm{D}$, then $[\![C_{(D/x)}]\!]^{\mathscr{M},v} = [\![C]\!]^{\mathscr{M},v(\mathrm{D}/x)}$.

Classical β-conversion rules

- **β-conversion rules** are *derivation rules* of STT by Church 1940:

Classical β-conversion rules

Let $C_{(D/x)}$ be C in which x is v-congruently *substituted* by D;
let $C_{(\bar{D}_m/\bar{x}_m)}$ be short for $C_{(D_1/x_1)\ldots(D_m/x_m)}$.

$$\frac{[\lambda\tilde{x}_m.C](\bar{D}_m) \qquad \textbf{β-redex}}{C_{(\bar{D}_m/\bar{x})} \qquad \textbf{β-contractum}} \ (\beta\text{-CON}_0)$$

('an application of a function to an argument leads to its value')

$$\frac{C_{(\bar{D}_m/\bar{x})}}{[\lambda\tilde{x}_m.C](\bar{D}_m)} \ (\beta\text{-EXP}_0)$$

where $D_1, x_1/\tau_1; \ldots; D_m, x_m/\tau_m; C/\tau$

Problem 1 – adequate rules of β-conversion

PROBLEM 1 (Adequate rules of β-conversion)

Are there adequate rules of β-conversion for PTT?

1. The classical rules aren't. (Proof below.)
2. We need their modified and adequate versions are stated/derived below.

Problem 1 – adequate rules of β-conversion

PROBLEM 1 (Adequate rules of β-conversion)

Are there adequate rules of β-conversion for PTT?

1. The classical rules aren't. (Proof below.)
2. We need their modified and adequate versions are stated/derived below.

Problem 1 – adequate rules of β-conversion

PROBLEM 1 (Adequate rules of β-conversion)

Are there adequate rules of β-conversion for PTT?

1. The classical rules aren't. (Proof below.)
2. We need their modified and adequate versions are stated/derived below.

Inadequacy of the classical rules of β-conversion

FACT 1 (Invalidity of the classical rules of β-conversion)

The classical rules of β-conversion do not generally hold in PTT.

Demonstration by a counter-example

Let $D, x/\tau_1; y/\tau_2; C/\tau_3$ Let D be v-improper and C contain a free occurrence of x. Is C_1 β-convertible to C_2?

$C_1 := [\lambda x.\lambda y.C](D)$ is v-*improper* because D is v-*improper*

$C_2 := [\lambda y.C]_{(D/x)}$ is (always) v-*proper*

Since, $C_1 \not\cong C_2$, $\cancel{(\beta\text{-CON}_0)}$, $\cancel{(\beta\text{-EXP}_0)}$

Example: $[\lambda x.\lambda y. \div(x,x)]\ (\div(3,0)) \not\rightarrow_\beta\ \lambda y. \div(\div(3,0), \div(3,0))$

Inadequacy of the classical rules of β-conversion

FACT 1 (Invalidity of the classical rules of β-conversion)

The classical rules of β-conversion do not generally hold in PTT.

Demonstration by a counter-example

Let $D, x/\tau_1; y/\tau_2; C/\tau_3$ Let D be v-improper and C contain a free occurrence of x. Is C_1 β-convertible to C_2?

$C_1 := [\lambda x.\lambda y.C](D)$ is v-*improper* because D is v-*improper*

$C_2 := [\lambda y.C]_{(D/x)}$ is (always) v-*proper*

Since, $C_1 \not\cong C_2$, ~~(β-CON$_0$)~~, ~~(β-EXP$_0$)~~

Example: $[\lambda x.\lambda y. \div (x, x)]\ (\div(3,0)) \not\rightarrow_\beta\ \lambda y. \div (\div(3,0), \div(3,0))$

Inadequacy of the classical rules of β-conversion

FACT 1 (Invalidity of the classical rules of β-conversion)

The classical rules of β-conversion do not generally hold in PTT.

Demonstration by a counter-example

Let $D, x/\tau_1; y/\tau_2; C/\tau_3$ Let D be v-improper and C contain a free occurrence of x. Is C_1 β-convertible to C_2?

$C_1 := [\lambda x.\lambda y.C](D)$ is v-*improper* because D is v-*improper*

$C_2 := [\lambda y.C]_{(D/x)}$ is (always) v-*proper*

Since, $C_1 \not\cong C_2$, ~~(β-CON$_0$)~~, ~~(β-EXP$_0$)~~

Example: $[\lambda x.\lambda y. \div (x, x)]\ (\div(3, 0)) \not\rightarrow_\beta\ \lambda y. \div (\div(3, 0), \div(3, 0))$

Content

Towards an appropriate ND for PTT

- Seeking an appropriate ND that and captures e.g. that D must be denoting
  - e.g. Tichý 1982, Moggi 1988, Farmer 1990

ND_{TT^*} – *natural deduction* in sequent style (adjusted from Tichý 1982)

i. embraces total and partial function
ii. captures *fine-grained hyperintensionality*
   e.g. $AbortiveComputation(\ulcorner \div(3,0) \urcorner)$
iii. resists Blamey's 1986 criticism (monotonicity of $\models$)
iv. *complete* w.r.t. (Henkin) *general models* (Kuchyňka 2020)

Towards an appropriate ND for PTT

- Seeking an appropriate ND that and captures e.g. that D must be denoting
  - e.g. Tichý 1982, Moggi 1988, Farmer 1990

ND_{TT^*} – *natural deduction* in sequent style (adjusted from Tichý 1982)

i. embraces total and partial function
ii. captures *fine-grained hyperintensionality*
   e.g. $AbortiveComputation(\ulcorner \div(3,0) \urcorner)$
iii. resists Blamey's 1986 criticism (monotonicity of $\models$)
iv. *complete* w.r.t. (Henkin) *general models* (Kuchyňka 2020)

Towards an appropriate ND for PTT

- Seeking an appropriate ND that and captures e.g. that D must be denoting
  - e.g. Tichý 1982, Moggi 1988, Farmer 1990

ND$_{\mathrm{TT}*}$ – *natural deduction* in sequent style (adjusted from Tichý 1982)

i. embraces total and partial function

ii. captures *fine-grained hyperintensionality*

e.g. $AbortiveComputation(\ulcorner \div(3,0) \urcorner)$

iii. resists Blamey's 1986 criticism (monotonicity of $\vDash$)

iv. *complete* w.r.t. (Henkin) *general models* (Kuchyňka 2020)

Towards an appropriate ND for PTT

- Seeking an appropriate ND that and captures e.g. that D must be denoting
  - e.g. Tichý 1982, Moggi 1988, Farmer 1990

ND$_{\mathrm{TT}*}$ – *natural deduction* in sequent style (adjusted from Tichý 1982)

i. embraces total and partial function

ii. captures *fine-grained hyperintensionality*

   e.g. $AbortiveComputation(\ulcorner \div(3,0) \urcorner)$

iii. resists Blamey's 1986 criticism (monotonicity of $\vDash$)

iv. *complete* w.r.t. (Henkin) *general models* (Kuchyňka 2020)

Towards an appropriate ND for PTT

- Seeking an appropriate ND that and captures e.g. that D must be denoting
  - e.g. Tichý 1982, Moggi 1988, Farmer 1990

ND$_{\mathsf{TT}*}$ – *natural deduction* in sequent style (adjusted from Tichý 1982)

i. embraces total and partial function
ii. captures *fine-grained hyperintensionality*
    e.g. $AbortiveComputation(\ulcorner \div(3,0) \urcorner)$
iii. resists Blamey's 1986 criticism (monotonicity of $\vDash$)
iv. *complete* w.r.t. (Henkin) *general models* (Kuchyňka 2020)

$\mathrm{ND}_{\mathrm{TT}^*}$: matches

Matches as bottom statements

Matches are *congruence statements* of the general form (where : is $\cong$)

$$\mathtt{M} := \ C : \underline{\mathbf{x}},$$

where C is either 'annotated' by i. _ (e.g. $=(x)$) or ii. $\mathbf{x}$

i. C:_ says that C is v-improper

ii. C:$\mathbf{x}$ says that C is v-proper and v-constructs

a. the object X v-constructed by the 'constant' $\mathbf{x}$

b. an object in the range of x

- v **satisfies** (in $\mathscr{M}$) $\mathtt{M}$ iff both $\mathtt{M}$'s C and $\underline{\mathbf{x}}$ v-construct the same/no object

$\mathrm{ND}_{\mathrm{TT}^*}$: matches

Matches as bottom statements

Matches are *congruence statements* of the general form (where : is $\cong$)

$$\mathtt{M} := \ C : \underline{\mathbf{x}},$$

where C is either 'annotated' by i. _ (e.g. $=(x)$) or ii. $\mathbf{x}$

i. C:_ says that C is v-improper

ii. C:$\mathbf{x}$ says that C is v-proper and v-constructs

   a. the object X v-constructed by the 'constant' $\mathbf{x}$

   b. an object in the range of x

- v **satisfies** (in $\mathscr{M}$) $\mathtt{M}$ iff both $\mathtt{M}$'s C and $\underline{\mathbf{x}}$ v-construct the same/no object

$\mathrm{ND}_{\mathrm{TT}^*}$: sequents and rules

$\mathrm{ND}_{\mathrm{TT}^*}$'s sequents and rules

Sequents are of the form (let Γ be a set of matches; $\longrightarrow$ is 'entailment')

$$\mathtt{S} := \quad \Gamma \longrightarrow \mathtt{M}$$

$\mathtt{S}$ is **valid** (in $\mathscr{M}$) iff every v that satisfies (in $\mathscr{M}$) all Γ's members satisfies (in $\mathscr{M}$) also $\mathtt{M}$.

Rules are validity-preserving operations of the form

$$\mathtt{R} := \quad \frac{\bar{\mathtt{S}}_m}{\mathtt{S}}$$

- *Examples:*

$$\frac{}{\Gamma, \mathtt{M} \longrightarrow \mathtt{M}}\ (\mathrm{AX}) \qquad \frac{\Gamma \longrightarrow \mathtt{M}}{\Gamma, \mathtt{M}' \longrightarrow \mathtt{M}}\ (\mathrm{WR})$$

$$\frac{\Gamma \longrightarrow \mathtt{M}_1 \qquad \Gamma \longrightarrow \mathtt{M}_2}{\Gamma \longrightarrow \mathtt{M}}\ (\mathrm{EFQ}) \qquad \frac{\Gamma, C{:}\_ \longrightarrow \mathtt{M} \qquad \Gamma, C{:}x \longrightarrow \mathtt{M}}{\Gamma \longrightarrow \mathtt{M}}\ (\mathrm{EXH})$$

Condition (EFQ): $\mathtt{M}_1$ and $\mathtt{M}_2$ are of the form $C{:}\underline{\mathtt{x}}$, but they don't v-construct the same object.

$\mathsf{ND}_{\mathsf{TT}^*}$: sequents and rules

$\mathsf{ND}_{\mathsf{TT}^*}$'s sequents and rules

Sequents are of the form (let Γ be a set of matches; $\longrightarrow$ is 'entailment')

$$\mathtt{S} := \quad \Gamma \longrightarrow \mathtt{M}$$

S is **valid** (in $\mathscr{M}$) iff every v that satisfies (in $\mathscr{M}$) all Γ's members satisfies (in $\mathscr{M}$) also M.

Rules are validity-preserving operations of the form

$$\mathtt{R} := \quad \frac{\bar{\mathtt{S}}_m}{\mathtt{S}}$$

- *Examples:*

$$\frac{}{\Gamma, \mathtt{M} \longrightarrow \mathtt{M}}\ (\mathsf{AX}) \qquad \frac{\Gamma \longrightarrow \mathtt{M}}{\Gamma, \mathtt{M}' \longrightarrow \mathtt{M}}\ (\mathsf{WR})$$

$$\frac{\Gamma \longrightarrow \mathtt{M}_1 \qquad \Gamma \longrightarrow \mathtt{M}_2}{\Gamma \longrightarrow \mathtt{M}}\ (\mathsf{EFQ}) \qquad \frac{\Gamma, C{:}\_ \longrightarrow \mathtt{M} \qquad \Gamma, C{:}x \longrightarrow \mathtt{M}}{\Gamma \longrightarrow \mathtt{M}}\ (\mathsf{EXH})$$

Condition (EFQ): $\mathtt{M}_1$ and $\mathtt{M}_2$ are of the form $C{:}\underline{\mathbf{x}}$, but they don't v-construct the same object.

Further three rules

- We'll employ also

$$\frac{}{\Gamma \longrightarrow \mathbf{x}{:}\mathbf{x}}\ \text{(TM)}$$

$$\frac{\Gamma \longrightarrow F(\bar{X}_m){:}\_ \qquad \Gamma \longrightarrow X_1{:}\mathbf{x}_1 \qquad \ldots \qquad \Gamma \longrightarrow X_m{:}\mathbf{x}_m}{\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_}\ (\text{app-SUB.i}^-)$$

$$\frac{\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_ \qquad \Gamma \longrightarrow X_1{:}\mathbf{x}_1 \qquad \ldots \qquad \Gamma \longrightarrow X_m{:}\mathbf{x}_m}{\Gamma \longrightarrow F(\bar{X}_m){:}\_}\ (\text{app-SUB.ii}^-)$$

- The rules (app-SUB^-) are *derivable* (similarly for (app-SUB.ii^-))

S	$\Gamma \longrightarrow F(\bar{X}_m){:}\_$	assumption in H
S_1	$\Gamma \longrightarrow X_1{:}\mathbf{x}_1$	assumption in H
$\vdots$	$\vdots$	$\vdots$
S_m	$\Gamma \longrightarrow X_m{:}\mathbf{x}_m$	assumption in H
1.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{\mathbf{x}}_m){:}\mathbf{y}$	(AX)
2.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{X}_m){:}\mathbf{y}$	from 1, $\bar{\text{S}}'_m$ by (app-SUB.ii)
3.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	from 2, S$'$ by (EFQ)
4.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\_ \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	(AX)
5.	$\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	from $3, 4$ by (EXH)

Further three rules

- We'll employ also

$$\frac{}{\Gamma \longrightarrow \mathbf{x}{:}\mathbf{x}}\ (\mathsf{TM})$$

$$\frac{\Gamma \longrightarrow F(\bar{X}_m){:}\_ \qquad \Gamma \longrightarrow X_1{:}\mathbf{x}_1 \qquad \dots \qquad \Gamma \longrightarrow X_m{:}\mathbf{x}_m}{\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_}\ (\text{app-SUB.i}^-)$$

$$\frac{\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_ \qquad \Gamma \longrightarrow X_1{:}\mathbf{x}_1 \qquad \dots \qquad \Gamma \longrightarrow X_m{:}\mathbf{x}_m}{\Gamma \longrightarrow F(\bar{X}_m){:}\_}\ (\text{app-SUB.ii}^-)$$

- The rules (app-SUB^-) are *derivable* (similarly for (app-SUB.ii^-))

S	$\Gamma \longrightarrow F(\bar{X}_m){:}\_$	assumption in H
S$_1$	$\Gamma \longrightarrow X_1{:}\mathbf{x}_1$	assumption in H
$\vdots$	$\vdots$	$\vdots$
S$_m$	$\Gamma \longrightarrow X_m{:}\mathbf{x}_m$	assumption in H
1.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{\mathbf{x}}_m){:}\mathbf{y}$	(AX)
2.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{X}_m){:}\mathbf{y}$	from 1, $\bar{\mathrm{S}}'_m$ by (app-SUB.ii)
3.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\mathbf{y} \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	from 2, S$'$ by (EFQ)
4.	$\Gamma, F(\bar{\mathbf{x}}_m){:}\_ \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	(AX)
5.	$\Gamma \longrightarrow F(\bar{\mathbf{x}}_m){:}\_$	from $3, 4$ by (EXH)

ND_{TT^*}'s primitive rules of β-conversion

The condition of the validity of the β-contraction-by-name rules

Each construction D_i, which we substitute in a λ-abstraction, must be v-proper.

- Tichý's 1982 β-rules immune to the above counter-example:

ND_{TT^*}'s primitive rules of β-conversion

$$\frac{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\mathbf{y}}{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{y}}\ (\beta\text{-CON})$$

$$\frac{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{y} \qquad \Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{x}_m}{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\mathbf{y}}\ (\beta\text{-EXP})$$

– the function $[\![\lambda \tilde{x}_m.C]\!]^{\mathscr{M},v}$ *is defined* for the m-tuple $[\![\bar{D}_m]\!]^{\mathscr{M},v}$

ND_{TT^*}'s primitive rules of β-conversion

The condition of the validity of the β-contraction-by-name rules

Each construction D_i, which we substitute in a λ-abstraction, must be v-proper.

- Tichý's 1982 β-rules immune to the above counter-example:

ND_{TT^*}'s primitive rules of β-conversion

$$\frac{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\mathbf{y}}{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{y}} \ (\beta\text{-CON})$$

$$\frac{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{y} \qquad \Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \dots \ \Gamma \longrightarrow D_m{:}\mathbf{x}_m}{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\mathbf{y}} \ (\beta\text{-EXP})$$

– the function $[\![\lambda \tilde{x}_m.C]\!]^{\mathscr{M},v}$ *is defined* for the m-tuple $[\![\bar{D}_m]\!]^{\mathscr{M},v}$

Problem 2 – β-conversion of constructions of undefined functions

- Cf. $[\lambda x. \div (x,0)]\ (3)\ \vdash\ \div(3,0)$, or our high-school example

PROBLEM 2 (β-conversion of constructions of undefined functions)

(β-CON) and (β-EXP) cannot handle abstractions v-constructing a function undefined for a given argument. Are there appropriate rules?

- We need additional, new β-conversion rules.

– *Subproblem*: Are they *derivable* from $\mathrm{ND}_{\mathrm{TT}*}$'s primitive rules?

Problem 2 – β-conversion of constructions of undefined functions

- Cf. $[\lambda x. \div (x,0)]\ (3)\ \vdash\ \div(3,0)$, or our high-school example

PROBLEM 2 (β-conversion of constructions of undefined functions)

(β-CON) and (β-EXP) cannot handle abstractions v-constructing a function undefined for a given argument. Are there appropriate rules?

- We need additional, new β-conversion rules.
  - *Subproblem*: Are they *derivable* from $\mathrm{ND}_{\mathrm{TT}*}$'s primitive rules?

Problem 2 – β-conversion of constructions of undefined functions

- Cf. $[\lambda x. \div (x, 0)]\ (3)\ \vdash\ \div(3, 0)$, or our high-school example

PROBLEM 2 (β-conversion of constructions of undefined functions)

(β-CON) and (β-EXP) cannot handle abstractions v-constructing a function undefined for a given argument. Are there appropriate rules?

- We need additional, new β-conversion rules.
  - *Subproblem*: Are they *derivable* from $\mathrm{ND}_{\mathrm{TT}^*}$'s primitive rules?

New rules of β-conversion are derivable

- To solve Problem 2,

Theorem 1: 'negative' variants of the β-conversion rules (β-CON) and (β-EXP)

$$\frac{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\_ \qquad \Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \ldots \ \Gamma \longrightarrow D_m{:}\mathbf{x}_m}{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_} \ (\beta\text{-CON}^-)$$

$$\frac{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_}{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\_} \ (\beta\text{-EXP}^-)$$

i. In both rules, the function $[\![\lambda \tilde{x}_m.C]\!]^{\mathscr{M},v}$ *is undefined* for the m-tuple $[\![\bar{D}_m]\!]^{\mathscr{M},v}$

ii. In (β-CON$^-$), $\Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ldots \Gamma \longrightarrow D_m{:}\mathbf{x}_m$ precludes the counter-example from Problem 1

New rules of β-conversion are derivable

- To solve Problem 2,

Theorem 1: 'negative' variants of the β-conversion rules (β-CON) and (β-EXP)

$$\frac{\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_ \qquad \Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{x}_m}{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_}\ (\beta\text{-CON}^-)$$

$$\frac{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_}{\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_}\ (\beta\text{-EXP}^-)$$

i. In both rules, the function $[\![\lambda\tilde{x}_m.C]\!]^{\mathscr{M},v}$ *is undefined* for the m-tuple $[\![\bar{D}_m]\!]^{\mathscr{M},v}$

ii. In (β-CON$^-$), $\Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ldots \Gamma \longrightarrow D_m{:}\mathbf{x}_m$ precludes the counter-example from Problem 1

New rules of β-conversion are derivable

- To solve Problem 2,

Theorem 1: 'negative' variants of the β-conversion rules (β-CON) and (β-EXP)

$$\frac{\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_ \qquad \Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{x}_m}{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_}\ (\beta\text{-CON}^-)$$

$$\frac{\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_}{\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_}\ (\beta\text{-EXP}^-)$$

i. In both rules, the function $[\![\lambda\tilde{x}_m.C]\!]^{\mathscr{M},v}$ *is undefined* for the m-tuple $[\![\bar{D}_m]\!]^{\mathscr{M},v}$

ii. In (β-CON$^-$), $\Gamma \longrightarrow D_1{:}\mathbf{x}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{x}_m$ precludes the counter-example from Problem 1

Proof of (β-CON$^-$)

S	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_$	assumption
S$_1$	$\Gamma \longrightarrow D_1{:}\mathbf{x}_1$	assumption
$\vdots$	$\vdots$	$\vdots$
S$_m$	$\Gamma \longrightarrow D_m{:}\mathbf{x}_m$	assumption
1.	$\Gamma, C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a} \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a}$	(AX)
2.	$\Gamma, C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a} \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a}$	from 1, $\bar{\text{S}}_m$ by (β-EXP)
3.	$\Gamma, C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a} \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_$	from S by (WR)
4.	$\Gamma, C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a} \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_$	from $2, 3$ by (EFQ)
5.	$\Gamma, C_{(\bar{D}_m/\bar{x}_m)}{:}\_ \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_$	(AX)
6.	$\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\_$	from $4, 5$ by (EXH)

Proof of (β-EXP$^-$)

S	$\Gamma \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}$:_	assumption
1.	$\Gamma, [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a} \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a}$	(AX)
2.	$\Gamma, [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a} \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}{:}\mathbf{a}$	from 1 by (β-CON)
3.	$\Gamma, [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a} \longrightarrow C_{(\bar{D}_m/\bar{x}_m)}$:_	from S by (WR)
4.	$\Gamma, [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a} \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m)$:_	from 2, 3 by (EFQ)
5.	$\Gamma, [\lambda\tilde{x}_m.C](\bar{D}_m)$:_ $\longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m)$:_	(AX)
6.	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m)$:_	from 4, 5 by (EXH)

Partial conclusion

- One specific quadruple of β-conversion rules

  a. (β-CON), (β-EXP) and (β-CON$^-$)
  – the substituted construction D_i is v-proper

  b. (β-CON$^-$) and (β-EXP$^-$)
  – relate v-improper constructions (their λ-abstractions v-construct an undefined function)

- We'll see another quadruple of β-conversion rules

  a. in all of them: the substituted construction D_i is v-proper

Partial conclusion

- One specific quadruple of β-conversion rules
  a. (β-CON), (β-EXP) and (β-CON$^-$)
  – the substituted construction D_i is v-proper
  b. (β-CON$^-$) and (β-EXP$^-$)
  – relate v-improper constructions (their λ-abstractions v-construct an undefined function)
- We'll see another quadruple of β-conversion rules
  a. in all of them: the substituted construction D_i is v-proper

Content

Towards β-conversion-by-value

- We want to substitute into a λ-abstracts not D as such, but its *'value'* – which must be *computed first*

– *call-by-value* evaluation in CS

PROBLEM 3 (Substitution by an already proven value)

Are there β-conversion rule that substitute the value of D which is an already proven object of a particular system?

Towards β-conversion-by-value

- We want to substitute into a λ-abstracts not D as such, but its *'value'* – which must be *computed first*

– *call-by-value* evaluation in CS

PROBLEM 3 (Substitution by an already proven value)

Are there β-conversion rule that substitute the value of D which is an already proven object of a particular system?

β-contraction by-value in computer science

- Usual formulation in CS (e.g Constable 1998); $\downarrow$ reads 'evaluates to' or 'is convergent to':

call-by-name rule

$$\frac{f \downarrow \lambda x.b \qquad b_{(z/x)} \downarrow c}{f(z) \downarrow c}$$

call-by-value rule

$$\frac{f \downarrow \lambda x.b \qquad a \downarrow a' \qquad b_{(a'/x)} \downarrow c}{f(a) \downarrow c}$$

New rules of β-conversion 'by-value'

- In $\mathrm{ND}_{\mathrm{TT}^*}$,

Theorem 2: Derived (and novel) β-conversion rules by-value

$$\frac{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\underline{\mathbf{a}} \qquad \Gamma \longrightarrow D_1{:}\mathbf{d}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{d}_m}{\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\underline{\mathbf{a}}}\ (\beta\text{-CON}^{V\pm})$$

$$\frac{\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\underline{\mathbf{a}} \qquad \Gamma \longrightarrow D_1{:}\mathbf{d}_1 \ \ldots\ \Gamma \longrightarrow D_m{:}\mathbf{d}_m}{\Gamma \longrightarrow [\lambda \tilde{x}_m.C](\bar{D}_m){:}\underline{\mathbf{a}}}\ (\beta\text{-EXP}^{V\pm})$$

- β-conversion provided it is first established that $\bar{D}_m$ result in values $\bar{\mathrm{d}}_m$ directly *acquired* by the respective acquisitions

Proofs of (β-CON^{V+}) and (β-CON^{V-})

Let $\bar{\mathbf{d}}_m/\bar{\tau}_m$ be not variables.

S_1	$\Gamma \longrightarrow D_1{:}\mathbf{d}_1$	assumption in H^{CON^V}
$\vdots$	$\vdots$	$\vdots$
S_m	$\Gamma \longrightarrow D_m{:}\mathbf{d}_m$	assumption in H^{CON^V}
$\mathsf{S}^{\mathrm{C}+}$	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a}$	assumption in H^{CON^V}
1.	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{\mathbf{d}}_m){:}\mathbf{a}$	from $\bar{\mathsf{S}}_m$ and $\mathsf{S}^{\mathrm{C}+}$ by (app-SUB.i)
CON^{V+}	$\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\mathbf{a}$	from 1 by (β-CON)
$\mathsf{S}^{\mathrm{C}-}$	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_$	assumption in H^{CON^V}
2.	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{\mathbf{d}}_m){:}\_$	from $\bar{\mathsf{S}}_m$ and $\mathsf{S}^{\mathrm{C}-}$ by (app-SUB.i$^-$)
CON^{V-}	$\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\_$	from $\bar{\mathsf{S}}_m$ and 2 by (β-CON$^-$)

Proofs of (β-EXP^{V+}) and (β-EXP^{V-})

S$_1$	$\Gamma \longrightarrow D_1{:}\mathbf{d}_1$	assumption in H^{EXP^V}
$\vdots$	$\vdots$	$\vdots$
S$_m$	$\Gamma \longrightarrow D_m{:}\mathbf{d}_m$	assumption in H^{EXP^V}
S$^{\mathrm{E}+}$	$\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\mathbf{a}$	assumption in H^{EXP^V}
3_1.	$\Gamma \longrightarrow \mathbf{d}_1{:}\mathbf{d}_1$	(TM)
$\vdots$	$\vdots$	$\vdots$
3_m.	$\Gamma \longrightarrow \mathbf{d}_m{:}\mathbf{d}_m$	(TM)
4.	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{\mathbf{d}}_m){:}\mathbf{a}$	from $\bar{3}_m$ and S$^{\mathrm{E}+}$ by (β-EXP)
EXP^{V+}	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\mathbf{a}$	from $\bar{\mathrm{S}}_m$ and 4 by (app-SUB.ii)
S$^{\mathrm{E}-}$	$\Gamma \longrightarrow C_{(\bar{\mathbf{d}}_m/\bar{x}_m)}{:}\_$	assumption in H^{EXP^V}
5.	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{\mathbf{d}}_m){:}\_$	from S$^{\mathrm{E}-}$ by (β-EXP$^-$)
EXP^{V-}	$\Gamma \longrightarrow [\lambda\tilde{x}_m.C](\bar{D}_m){:}\_$	from $\bar{\mathrm{S}}_m$ and 5 by (app-SUB.ii$^-$)

Content

Conclusions

- In TT* which is an expressive system with both total and partial functions we

1. showed *appropriate rules* of β-conversion by-name and by-value
   - because STT's classical rules fail in PTT (Fact 1)
2. formulated *novel* ('negative') *rules* β-conversion by-name and by-value
   - because rules such as (β-CON) and (β-EXP) do not handle an application of an undefined function
3. *derived* all the rules from the initial couple of β-conversion by-name rules (Theorem 1 and Theorem 2)
   - which shows superiority of (β-CON) and (β-EXP)

Conclusions

- In TT* which is an expressive system with both total and partial functions we

1. showed *appropriate rules* of β-conversion by-name and by-value
   - because STT's classical rules fail in PTT (Fact 1)
2. formulated *novel* ('negative') *rules* β-conversion by-name and by-value
   - because rules such as (β-CON) and (β-EXP) do not handle an application of an undefined function
3. *derived* all the rules from the initial couple of β-conversion by-name rules (Theorem 1 and Theorem 2)
   - which shows superiority of (β-CON) and (β-EXP)

Conclusions

- In TT* which is an expressive system with both total and partial functions we

1. showed *appropriate rules* of β-conversion by-name and by-value
   - because STT's classical rules fail in PTT (Fact 1)
2. formulated *novel* ('negative') *rules* β-conversion by-name and by-value
   - because rules such as (β-CON) and (β-EXP) do not handle an application of an undefined function
3. *derived* all the rules from the initial couple of β-conversion by-name rules (Theorem 1 and Theorem 2)
   - which shows superiority of (β-CON) and (β-EXP)

Conclusions

- In TT* which is an expressive system with both total and partial functions we

1. showed *appropriate rules* of β-conversion by-name and by-value
   - because STT's classical rules fail in PTT (Fact 1)
2. formulated *novel* ('negative') *rules* β-conversion by-name and by-value
   - because rules such as (β-CON) and (β-EXP) do not handle an application of an undefined function
3. *derived* all the rules from the initial couple of β-conversion by-name rules (Theorem 1 and Theorem 2)
   - which shows superiority of (β-CON) and (β-EXP)

References & Thank you!

Kuchyňka, P.; Raclavský, J. (2021): Rules of β-conversion-by-name, β-conversion-by-value, and η-conversion in partial type theory. *Logic Journal of the IGPL*. conditionally accepted