Extensional and Intensional Semantic Universes: A Denotational Model of Dependent Types

Valentin Blot

INRIA, LMF
Université Paris-Saclay

Jim Laird

University of Bath

Denotational semantics

- Fully abstract semantics for PCF
  - Scott domains
    no full abstraction (parallel or)
  - stable functions
    no full abstraction (Gustave)
  - sequential algorithms
    full abstraction for SPCF
  - game semantics
    full abstraction through an extensional collapse
- Semantics for dependent types
  - Scott domains (Palmgren - Stoltenberg-Hansen)
  - game semantics (Vákár, Abramsky, Jagadeesan)
  - stable functions + sequential algorithms (this talk)

Two semantic universes

Intensional

Types

concrete data structures

cells (opponent moves)

values (player moves)

Terms

sequential algorithms

computation strategies

Extensional

Types

dI-domains

particular Scott domains

Terms

stable functions

particular cont. functions

extensional quotient

$+$ a dI-domain $\mathcal{I}$ of *all* concrete data structures

extensional terms in $\mathcal{I}$ are intensional types

The intensional universe

Intensional dependent types in game semantics

Example: $\Pi\,(n : \mathbf{nat})\,.\mathbf{vec}\,(n)$

opponent player

$\Pi\,(n : \mathbf{nat}) \quad \mathbf{vec}\,(n)$

$?_i$

Intensional dependent types in game semantics

Example: $\Pi\,(n : \mathbf{nat})\,.\mathbf{vec}\,(n)$

opponent player

$\Pi\,(n : \mathbf{nat}) \quad \mathbf{vec}\,(n)$

$?_i$

$?$

Intensional dependent types in game semantics

Example: $\Pi\,(n : \mathbf{nat})\,.\mathbf{vec}\,(n)$

opponent player

$\Pi\,(n : \mathbf{nat}) \quad \mathbf{vec}\,(n)$

$?_i$

$?$

$i - 1$

Intensional dependent types in game semantics

Example: $\Pi\,(n : \mathbf{nat})\,.\mathbf{vec}\,(n)$

		opponent	player
$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$		
	$?_i$		
$?$			
$i - 1$			
	what!?		

Intensional dependent types in game semantics

Example: $\Pi\,(n : \mathbf{nat})\,.\mathbf{vec}\,(n)$

opponent player

$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
	$?_i$
$?$	
$i - 1$	
	what!?

- Opponent move $?_i$ should restrict plays on the left
- Game semantics is too intensional

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

$?$

Game semantics vs. sequential algorithms

Game semantics

$(\textbf{nat} \to \textbf{nat}) \to \textbf{nat}$

$$\begin{array}{ccc} & & ? \\ & ? & \\ ? & & \\ n & & \\ & m & \end{array}$$

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

$?$

$?$

$?$

n

m

n

m'

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

$?$

$?$

$?$

n

m

n

m'

p

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

		?
	?	
?		
n		
	m	
n		
	m'	
		p

Sequential algorithms

arg $\quad (\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

$\bot$ $\quad$?

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to$	$\mathbf{nat})$	$\to \mathbf{nat}$
		?
	?	
?		
n		
	m	
n		
	m'	
		p

Sequential algorithms

arg	$(\mathbf{nat} \to$	$\mathbf{nat})$	$\to \mathbf{nat}$
$\bot$			?
		?	
$\{\}$	?		

Game semantics vs. sequential algorithms

Game semantics		
$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$		
		$?$
	$?$	
$?$		
n		
	m	
n		
	m'	
		p

Sequential algorithms			
arg	$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$		
$\perp$			$?$
		$?$	
$\{\}$	$?$		
	n		
$\{n \mapsto m\}$		m	

Game semantics vs. sequential algorithms

Game semantics

$(\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat}$

		?
	?	
?		
n		
	m	
n		
	m'	
		p

Sequential algorithms

arg	$(\mathbf{nat}$	$\to \mathbf{nat})$	$\to \mathbf{nat}$
$\perp$			?
		?	
$\{\}$	?		
	n		
$\{n \mapsto m\}$		m	
	$\not n$		

Game semantics vs. sequential algorithms

Game semantics

($\mathbf{nat}$	$\to \mathbf{nat}$)	$\to \mathbf{nat}$
		?
	?	
?		
n		
	m	
n		
	m'	
		p

Sequential algorithms

arg	($\mathbf{nat}$	$\to \mathbf{nat}$)	$\to \mathbf{nat}$
$\perp$			?
		?	
$\{\}$	?		
	n		
$\{n \mapsto m\}$		m	
	$\not{n}$		
			p

Game semantics vs. sequential algorithms

Game semantics

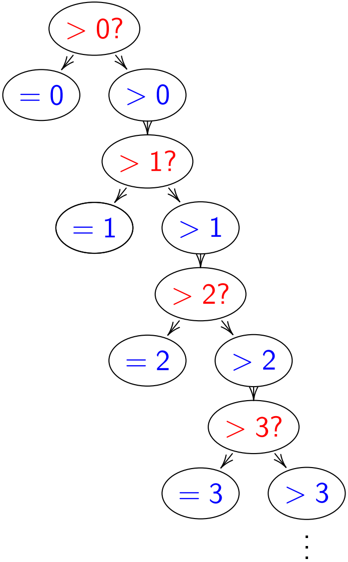
$(\mathbf{nat}$	$\to \mathbf{nat})$	$\to \mathbf{nat}$
		?
	?	
?		
n		
	m	
n		
	m'	
		p

Sequential algorithms

arg	$(\mathbf{nat}$	$\to \mathbf{nat})$	$\to \mathbf{nat}$
$\bot$			?
		?	
$\{\}$	?		
	n		
$\{n \mapsto m\}$		m	
	$\not{n}$		
			p

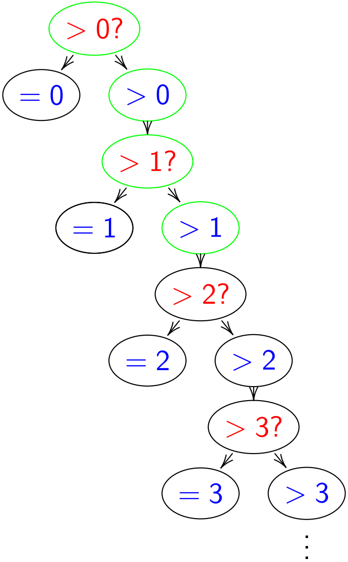
- Sequential algorithms are constrained by nature
- The argument grows monotonically, step by step

Intensional dependent types in sequential algorithms



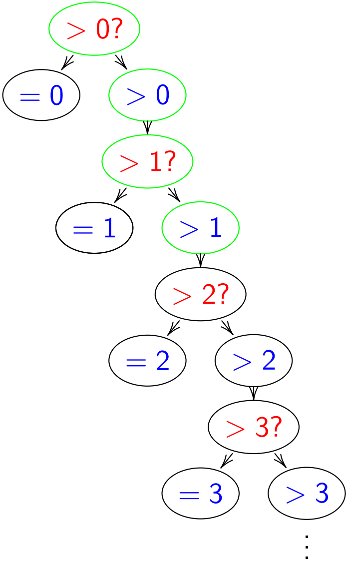
arg $\quad \Pi\,(n : \mathbf{nat}) \quad \mathbf{vec}\,(n)$

Intensional dependent types in sequential algorithms



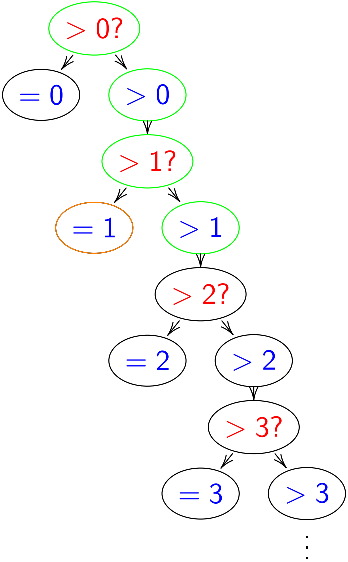
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$

Intensional dependent types in sequential algorithms



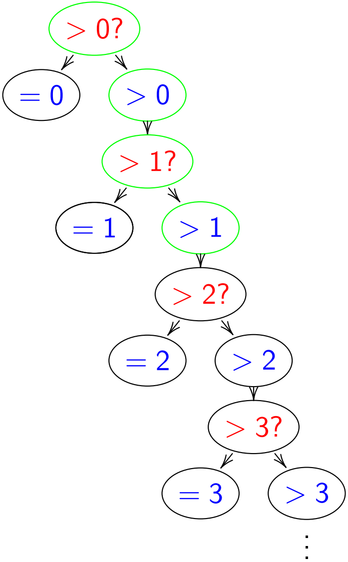
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	

Intensional dependent types in sequential algorithms



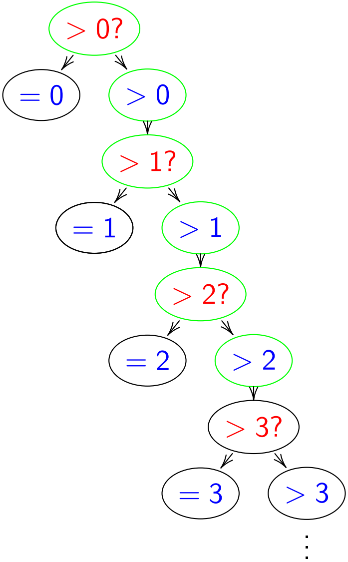
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	
	$> 1?$	
> 1	~~$= 1$~~	

Intensional dependent types in sequential algorithms



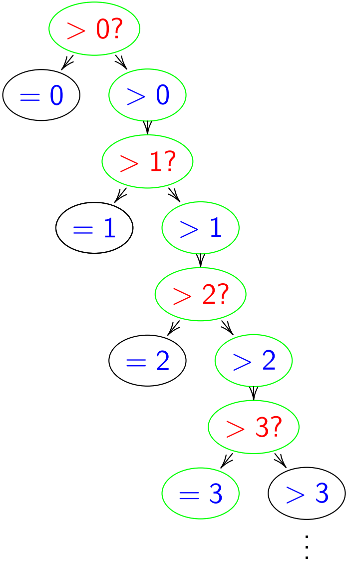
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	
	$> 1?$	
> 1	~~$= 1$~~	
> 1	> 1	

Intensional dependent types in sequential algorithms



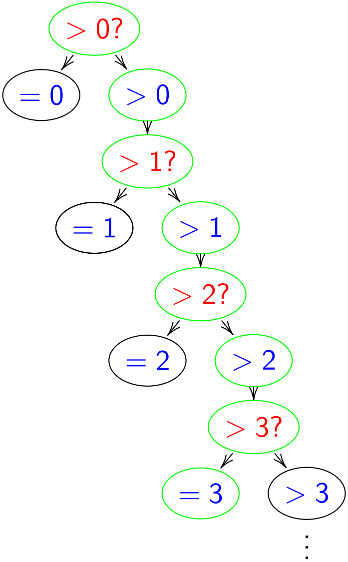
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	
	$> 1?$	
> 1	~~$= 1$~~	
> 1	> 1	
	$> 2?$	
> 2	> 2	

Intensional dependent types in sequential algorithms



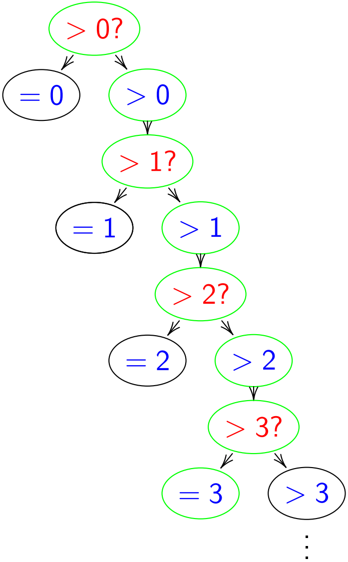
arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
>1		$?_1$
	$>0?$	
>1	>0	
	$>1?$	
>1	~~$=1$~~	
>1	>1	
	$>2?$	
>2	>2	
	$>3?$	
$=3$	$=3$	

Intensional dependent types in sequential algorithms



arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	
	$> 1?$	
> 1	~~$= 1$~~	
> 1	> 1	
	$> 2?$	
> 2	> 2	
	$> 3?$	
$= 3$	$= 3$	
		$()$

Intensional dependent types in sequential algorithms



arg	$\Pi\,(n : \mathbf{nat})$	$\mathbf{vec}\,(n)$
> 1		$?_1$
	$> 0?$	
> 1	> 0	
	$> 1?$	
> 1	~~$= 1$~~	
> 1	> 1	
	$> 2?$	
> 2	> 2	
	$> 3?$	
$= 3$	$= 3$	
		$()$

- Opponent move $?_1$ restricts the argument to be > 1

The extensional universe

A model of the extensional universe

Refinement of Palmgren - Stoltenberg-Hansen's

- The class $\mathcal{DOM}$ of dI-domains is a dI-domain.
- $\mathcal{DOM}$-parametrization:

$$F : D \to \mathcal{DOM} \text{ stable}$$

$$\text{(where } D \text{ is a dI-domain)}$$

- Dependent stable function on F:

$$f : D \to \bigcup_{x \in D} F(x) \text{ stable with } f(x) \in F(x)$$

these form a dI-domain $\Pi(D, F)$

- Dependent pair on F:

$$p \in D \times \bigcup_{x \in D} F(x) \text{ with } \pi_2(f) \in F(\pi_1(f))$$

these form a dI-domain $\Sigma(D, F)$

Relating the two universe

Categories with families

A category with families is:

- a functor :

$$\begin{array}{llll} \mathcal{F}: & \mathcal{C}^{op} & \to & \mathrm{Fam} \\ & \Gamma & \mapsto & (\mathit{Term}(\Gamma, T))_{T \in \mathit{Type}(\Gamma)} \end{array}$$

  where Fam is the category of set-indexed families of sets

- a terminal object (empty context) in $\mathcal{C}$
- a context extension operation:
  if $\Gamma \in \mathcal{C}$ and $T \in \mathit{Type}(\Gamma)$ then $\Gamma.T \in \mathcal{C}$ such that for all...

A category with families is a model of type theory

Intensional and extensional categories with families

$$\begin{aligned} \mathcal{C}_{\mathcal{I}}^{op} &\rightarrow \mathrm{Fam} \\ M &\mapsto (Term_{\mathcal{I}}(M, A))_{A \in Type_{\mathcal{I}}(M)} \end{aligned}$$

- $\mathcal{C}_{\mathcal{I}}$:
  - objects:
    concrete data structures
  - morphisms:
    sequential algorithms
- $Type_{\mathcal{I}}(M)$:
  $A : D(M) \rightarrow \mathcal{CDS}$ stable
- $Term_{\mathcal{I}}(M, A)$:
  dependent seq. algorithms

$$\begin{aligned} \mathcal{C}_{\mathcal{E}}^{op} &\rightarrow \mathrm{Fam} \\ D &\mapsto (Term_{\mathcal{E}}(D, F))_{F \in Type_{\mathcal{E}}(D)} \end{aligned}$$

- $\mathcal{C}_{\mathcal{E}}$:
  - objects:
    dI-domains
  - morphisms:
    stable functions
- $Type_{\mathcal{E}}(D)$:
  $F : D \rightarrow \mathcal{DOM}$ cont.
- $Term_{\mathcal{I}}(D, F)$:
  dependent stable functions

Intensional and extensional categories with families

$$\begin{aligned}\mathcal{C}_{\mathcal{I}}^{op} &\to \mathrm{Fam} \\ M &\mapsto (Term_{\mathcal{I}}(M,A))_{A\in Type_{\mathcal{I}}(M)}\end{aligned}$$

- $\mathcal{C}_{\mathcal{I}}$:
  - objects:
    concrete data structures
  - morphisms:
    sequential algorithms
- $Type_{\mathcal{I}}(M)$:
  $A : D(M) \to \mathcal{CDS}$ stable
- $Term_{\mathcal{I}}(M, A)$:
  dependent seq. algorithms

$$\begin{aligned}\mathcal{C}_{\mathcal{E}}^{op} &\to \mathrm{Fam} \\ D &\mapsto (Term_{\mathcal{E}}(D,F))_{F\in Type_{\mathcal{E}}(D)}\end{aligned}$$

- $\mathcal{C}_{\mathcal{E}}$:
  - objects:
    dI-domains
  - morphisms:
    stable functions
- $Type_{\mathcal{E}}(D)$:
  $F : D \to \mathcal{DOM}$ cont.
- $Term_{\mathcal{I}}(D, F)$:
  dependent stable functions

Universe hierarchy: extensional terms are intensional types

Extensional quotient

$$\begin{array}{rccl} \mathcal{F}_{\mathcal{I}}: & \mathcal{C}_{\mathcal{I}}^{op} & \to & \mathrm{Fam} \\ & M & \mapsto & (Term_{\mathcal{I}}(M,A))_{A \in Type_{\mathcal{I}}(M)} \end{array}$$

$$\begin{array}{rccl} \mathcal{F}_{\mathcal{E}}: & \mathcal{C}_{\mathcal{E}}^{op} & \to & \mathrm{Fam} \\ & D & \mapsto & (Term_{\mathcal{E}}(D,F))_{F \in Type_{\mathcal{E}}(D)} \end{array}$$

$G : \mathcal{C}_{\mathcal{I}} \to \mathcal{C}_{\mathcal{E}}$ sends:

- a CDS M to the dI-domain $D(M)$ of its states
- a sequential algorithm $a : M \to N$ to the function it computes $\mathrm{fun}(a) : D(M) \to D(N)$

$\phi : \mathcal{F}_{\mathcal{I}} \to \mathcal{F}_{\mathcal{E}} \circ G^{op}$ natural transformation s.t. ϕ_M sends:

- $A : D(M) \to \mathcal{CDS}$ to $G \circ A : D(M) \to \mathcal{DOM}$
- dependent seq. alg. a to dependent stable function $\mathrm{fun}(a)$

Universe cumulativity: any term/type in $\mathcal{I}$ can be lifted to $\mathcal{E}$

A type theory with two universes

- Intensional and extensional universes

$$\Gamma \vdash_{\mathcal{I}} t : T \qquad \Gamma \vdash_{\mathcal{E}} t : T \qquad \frac{\Gamma \vdash_{\mathcal{E}} T : \mathcal{I}}{\Gamma \vdash_{\mathcal{I}} T \text{ type}}$$

- Cumulativity

$$\frac{\Gamma \vdash_{\mathcal{I}} t : T}{\Gamma \vdash_{\mathcal{E}} t : T}$$

- Dependent products and sums

$$\Pi_{\mathcal{U}}\,(x : S)\,.\,T \text{ type} \qquad \Sigma_{\mathcal{U}}\,(x : S)\,.\,T \text{ type} \qquad \text{for } \underline{\mathcal{U} \in \{\mathcal{I}, \mathcal{E}\}}$$

- Booleans

$$\frac{}{\vdash_{\mathcal{U}} \mathtt{tt}, \mathtt{ff} : \mathbf{bool}} \qquad \frac{\Gamma, x : \mathbf{bool} \vdash_{\mathcal{U}} T \text{ type} \qquad \Gamma \vdash_{\mathcal{U}} t_1 : T\,[\mathtt{tt}/x] \ldots}{\Gamma \vdash_{\mathcal{U}} \mathtt{If}\ s\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2 : T\,[s/x]}$$

- General recursion

$$\frac{\Gamma, x : T \vdash_{\mathcal{U}} t : T}{\Gamma \vdash_{\mathcal{U}} \mu x.t : T}$$

(and therefore recursive types)

Expressivity

- Function types:

$$T \to U ::= \Pi\left(x : T\right).U \text{ if } x \notin FV\left(U\right)$$

- Product types:

$$T \times U ::= \Sigma\left(x : T\right).U \text{ if } x \notin FV\left(U\right)$$

- Unit:

$$\mathbf{1} ::= \mu x : \mathcal{I}.x$$

- Disjoint sum:

$$T \oplus U ::= \Sigma\left(x : \mathbf{bool}\right).\texttt{If}\ x\ \texttt{then}\ T\ \texttt{else}\ U$$

- Natural numbers:

$$\mathbf{nat} ::= \mu x : \mathcal{I}.\mathbf{1} \oplus x$$

- Vectors of booleans:

$$\begin{aligned}\mathbf{vec} ::= \ & \mu f : \mathbf{nat} \to \mathcal{I}. \\ & \lambda x : \mathbf{nat}.\texttt{If}\ \pi_1\left(x\right)\ \texttt{then}\ \mathbf{1}\ \texttt{else}\ \mathbf{B} \times f\left(\pi_2\left(x\right)\right)\end{aligned}$$

A programming language for the intensional universe

We define a straightforward operational semantics and we get:

Theorem (Computational adequacy)

If $\vdash_{\mathcal{I}} t : \mathbf{bool}$ *then* $t \Downarrow \mathtt{tt} \iff [t]_{\mathcal{I}} = [\mathtt{tt}]_{\mathcal{I}}$

For full completeness of the finite fragment and for full abstraction we need to extend our language:

$$\frac{\Gamma, k : \mathbf{bool} \vdash_{\mathcal{I}} t : \mathbf{bool}}{\Gamma \vdash_{\mathcal{I}} \mathtt{catch}\,(k)\,.t : \mathbf{bool}}$$

$$\mathtt{catch}\,(k)\,.E\,[k] \to \mathtt{tt} \qquad \mathtt{catch}\,(k)\,.v \to \mathtt{ff}$$

$$\begin{aligned} t = \mathtt{If}\ \ \mathtt{catch}\,(k)\,.t\ \ &\mathtt{then}\ \ (\mathtt{If}\ \ k\ \ \mathtt{then}\ \ t\,[\mathtt{tt}/k]\ \ \mathtt{else}\ \ t\,[\mathtt{ff}/k]) \\ &\mathtt{else}\ \ t\,[\mathtt{tt}/k] \end{aligned}$$

$t = \mathtt{If}\ \ s\ \ \mathtt{then}\ \ t\ \ \mathtt{else}\ \ t$, provided $t : \mathtt{If}\ \ s\ \ \mathtt{then}\ \ T\ \ \mathtt{else}\ \ T$

Full completeness and full abstraction

Finite total fragment: no recursion except $\mathbf{1} ::= \mu x : \mathcal{I}.x$
We have full completeness:

Theorem (Full completeness)

For finite $\Gamma \vdash_{\mathcal{I}} T$ *type and total* $x \in [T]^{\Gamma}_{\mathcal{I}}$ *there exists a finite term* $\Gamma \vdash_{\mathcal{I}} t_x : T$ *with* $[t_x]^{\Gamma}_{\mathcal{I}} = x$.

We obtain finite definability in the full theory and therefore full abstraction:

Theorem (Full abstraction)

If $\Gamma \vdash_{\mathcal{I}} t_1, t_2 : T$ *then* $t_1 \lesssim^{\Gamma}_{T} t_2 \iff [t_1]^{\Gamma}_{\mathcal{I}} \subseteq [t_2]^{\Gamma}_{\mathcal{I}}$

Towards identity types

For M a concrete data structure:

$$\begin{array}{rcccccl} Eq: & D(M) & \times & D(M) & \to & \mathcal{CDS} \\ & (x & , & y) & \mapsto & "x \cap y" \end{array}$$

- $x \in \mathtt{Eq}(x, x)$.
- If $x \cup y \in D(M)$ then $\mathtt{Eq}(x, y)$ is the down-closure of $x \cap y \in D(M)$. In particular we can have $\mathtt{Eq}(x, y) \neq \{\bot\} = [\mu x.x]$.
- If x and y are total then $\mathtt{Eq}(x, y)$ contains a total element if and only if $x = y$.