What Is More Fundamental: Partial Functions or Types?

Đorđe Marković

University of Klagenfurt
EX3 Project - Alice Tarzariol

Continued PhD project - Marc Denecker

20. 5. 2026 – Online – Potassco Seminar

A Prudent Logic of Partial Functions

vs

Order-Sorted Logic

A Prudent Logic of Partial Functions

vs

Order-Sorted Logic

A Prudent Logic of Partial Functions

vs

Order-Sorted Logic

Partial Functions

Partial Functions

The present King of France is bald.

Partial Functions

The present King of France is bald.

Bertrand Russell

There exists an x which is the king of France and x is bald.

Partial Functions

The present King of France is bald.

Bertrand Russell

There exists an x which is the king of France and x not is bald.

Partial Functions

The present King of France is bald.

Bertrand Russell

There does not exists an x which is the king of France and x is bald.

Partial Functions

The present King of France is bald.

Bertrand Russell

There exists an x which is the king of France and x is bald.

Peter Strawson

Such a sentence does not have a truth value!

Partial Functions

The present King of France is bald.

KRR perspective: Such a sentence is ambiguous!

Partial Functions

The present King of France is bald.

KRR perspective: Such a sentence is ambiguous!

There exists an x which is the king of France and x is bald.

Partial Functions

The present King of France is bald.

KRR perspective: Such a sentence is ambiguous!

There exists an x which is the king of France and x is bald.

For all x, if x is the king of France then x is bald.

A Prudent Logic of Partial Functions FO(PF)

A Prudent Logic of Partial Functions

The present King of France is bald.

$$Bald(KingOf(France))$$

A Prudent Logic of Partial Functions

The present King of France is bald.

$$Bald(KingOf(France))$$

The present king of France exists.

$$\delta_{KingOf}(France)$$

The present King of France exists and he is bald.

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

$$\textbf{if } \delta_{KingOf}(France) \textbf{ then } Bald(KingOf(France)) \textbf{ else } false$$

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

if $\delta_{KingOf}(France)$ **then** $Bald(KingOf(France))$ **else** $false$

If the present King of France exists, then he is bald.

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

if $\delta_{KingOf}(France)$ **then** $Bald(KingOf(France))$ **else** $false$

If the present King of France exists, then he is bald.

if $\delta_{KingOf}(France)$ **then** $Bald(KingOf(France))$ **else** $true$

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

$$\delta_{KingOf}(France) \overrightarrow{\wedge} Bald(KingOf(France))$$

If the present King of France exists, then he is bald.

if $\delta_{KingOf}(France)$ **then** $Bald(KingOf(France))$ **else** $true$

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

$$\delta_{KingOf}(France) \overrightarrow{\wedge} Bald(KingOf(France))$$

If the present King of France exists, then he is bald.

$$\delta_{KingOf}(France) \Rightarrow Bald(KingOf(France))$$

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

$$[[Bald(KingOf(France))]]$$

If the present King of France exists, then he is bald.

$$\delta_{KingOf}(France) \rightrightarrows Bald(KingOf(France))$$

A Prudent Logic of Partial Functions

The present King of France exists and he is bald.

$$[[Bald(KingOf(France))]]$$

If the present King of France exists, then he is bald.

$$\langle\langle Bald(KingOf(France))\rangle\rangle$$

A Prudent Logic of Partial Functions

$$f(x)$$

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

$\overrightarrow{\wedge}$ $\rightrightarrows$ $[\![\cdot]\!]$ $\langle\!\langle \cdot \rangle\!\rangle$

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

$\overrightarrow{\wedge}$ $\Rrightarrow$ $[\![\cdot]\!]$ $\langle\!\langle \cdot \rangle\!\rangle$

Well-guarded formula

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

$\overrightarrow{\wedge}$ $\rightrightarrows$ $[\![\cdot]\!]$ $\langle\langle \cdot \rangle\rangle$

Non-ambiguous formula

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

$\overrightarrow{\wedge}$ $\rightrightarrows$ $[\![\cdot]\!]$ $\langle\!\langle \cdot \rangle\!\rangle$

Polynomially Decidable

A Prudent Logic of Partial Functions

$f(x)$ $\delta_f(x)$

if $\delta_f(x)$ **then** $\ldots f(x) \ldots$ **else** χ

$\overrightarrow{\wedge}$ $\rightrightarrows$ $[\![\cdot]\!]$ $\langle\langle \cdot \rangle\rangle$

Well-defined formula

Term guarding

Term guarding

$KingOf(\underline{\text{France}})$

A Prudent Logic of Partial Functions

Term guarding

$$KingOf(\underline{France})$$

$$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$$

A Prudent Logic of Partial Functions

Term guarding

$$KingOf(\underline{France})$$

$$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$$

$$\Delta(KingOf(France))$$

A Prudent Logic of Partial Functions

Term guarding

$$KingOf(\underline{France})$$

$$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$$

$$\Delta(KingOf(France))$$

Indirect guarding

A Prudent Logic of Partial Functions

Term guarding

$KingOf(\underline{France})$

$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$

$\Delta(KingOf(France))$

Indirect guarding

$\delta_{KingOf}(France).$

A Prudent Logic of Partial Functions

Term guarding

$KingOf(\underline{France})$

$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$

$\Delta(KingOf(France))$

Indirect guarding

$\delta_{KingOf}(France).$

$Bald(KingOf(France)).$

A Prudent Logic of Partial Functions

Term guarding

$KingOf(\underline{France})$

$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$

$\Delta(KingOf(France))$

Indirect guarding

$\delta_{KingOf}(France).$

$Bald(KingOf(France)).$

Total functions

A Prudent Logic of Partial Functions

Term guarding

$KingOf(\underline{France})$

$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$

$\Delta(KingOf(France))$

Indirect guarding

$\delta_{KingOf}(France).$

$Bald(KingOf(France)).$

Total functions

$\forall x : \delta_{KingOf}(x).$

A Prudent Logic of Partial Functions

Term guarding

$KingOf(\underline{France})$

$\delta_{France} \overrightarrow{\wedge} \delta_{KingOf}(France)$

$\Delta(KingOf(France))$

Partial predicates

Indirect guarding

$\delta_{KingOf}(France).$

$Bald(KingOf(France)).$

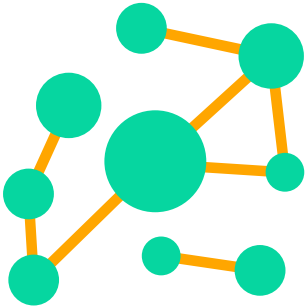
Total functions

$\forall x : \delta_{KingOf}(x).$

Defining Partial Functions

Recursive definitions

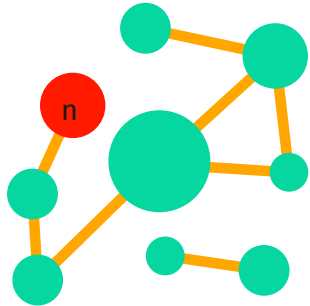
Distance function



Defining Partial Functions

Recursive definitions

Distance function

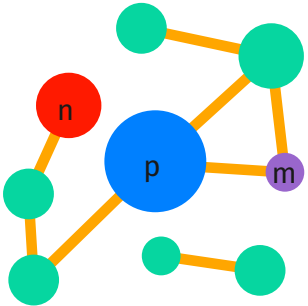


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \\ \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

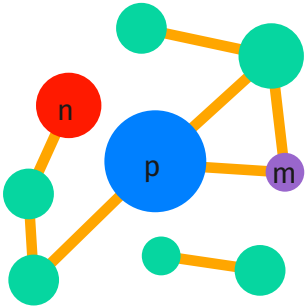


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = \end{array} \right.$$

Defining Partial Functions

Recursive definitions

Distance function

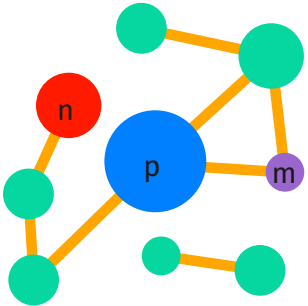


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

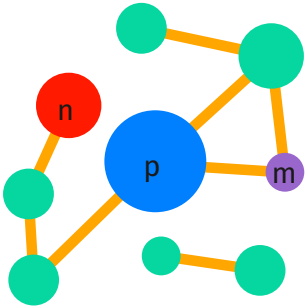


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

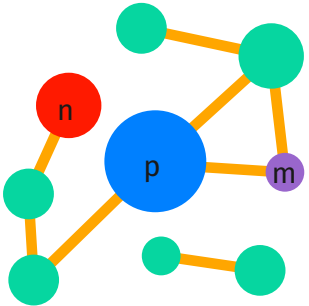


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \wedge (n \neq m) \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

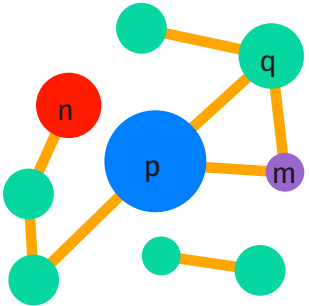


$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \wedge (n \neq m) \wedge E(p, m) \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

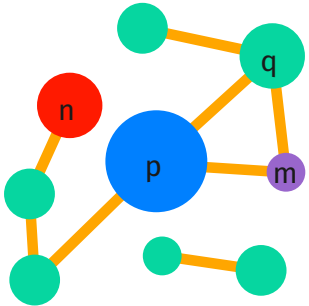


$$\left\{ \begin{array}{l} \forall n : dist(n,n) = 0 \\ \forall n,m,p : dist(n,m) = dist(n,p) + 1 \leftarrow \Delta(dist(n,p)+1) \wedge (n \neq m) \wedge E(p,m) \\ \qquad\qquad \wedge \neg\exists q : E(q,m) \wedge [[d(n,q) < d(n,p)]] \end{array} \right\}$$

Defining Partial Functions

Recursive definitions

Distance function

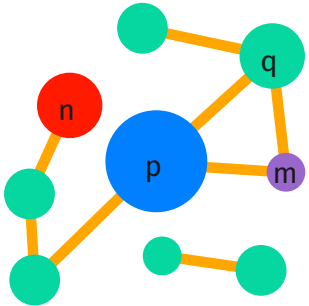


$$\{\ \forall x : f(t) = k \leftarrow \Delta(t) \wedge \Delta(k) \wedge \phi.\ \}$$

Defining Partial Functions

Recursive definitions

Distance function

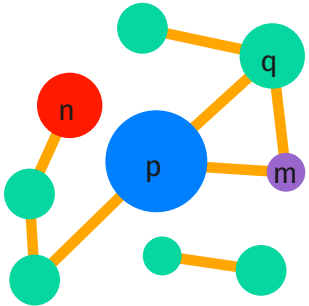


$$\{\ \forall x : f(t) = k \leftarrow \Delta(t) \wedge \Delta(k) \wedge \phi.\ \}$$

Defining Partial Functions

Recursive definitions

Distance function

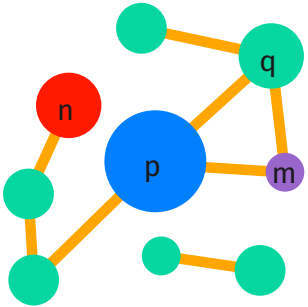


$$\{\ \forall x : f(t) = k \leftarrow \Delta(t) \wedge \Delta(k) \wedge \phi.\ \}$$

Defining Partial Functions

Recursive definitions

Distance function

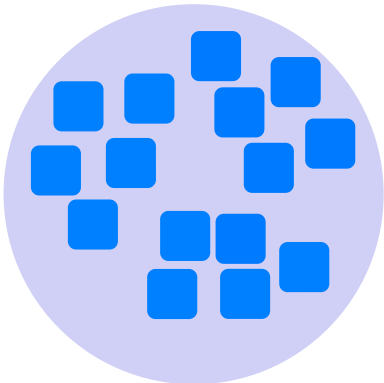


$$\{\ \forall x : f(t) = k \leftarrow \Delta(t) \wedge \Delta(k) \wedge \phi.\ \}$$

A prudent logic of partial functions, Đ. Marković, R. Van den Eede, M. Denecker,
Annals Of Mathematics And Artificial Intelligence (2026)

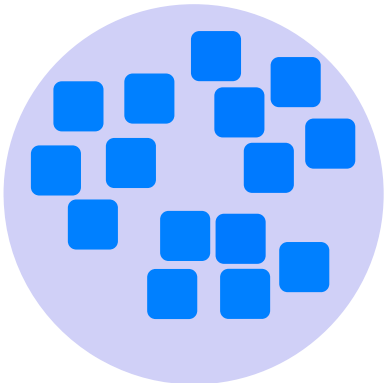
Order-sorted Logic FO(OS)

Classical (Untypesd) Logic

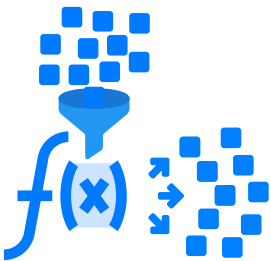


Untyped Domain

Classical (Untypesd) Logic

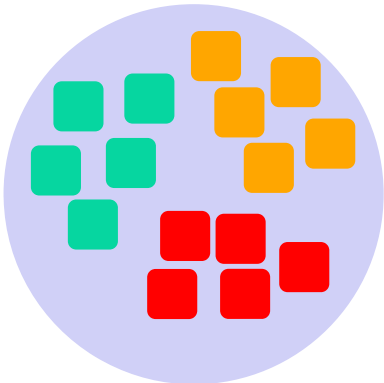


Untyped Domain

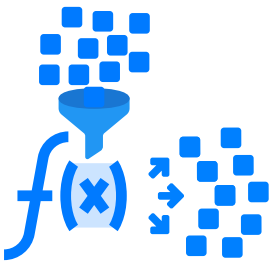


Untyped Functions

Many-Sorted Logic

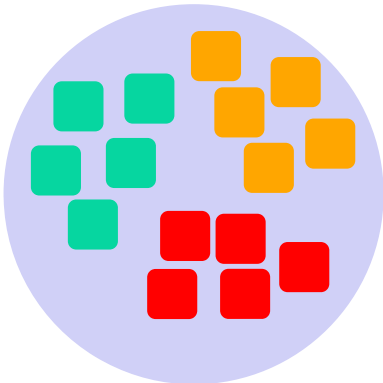


Typed Domain

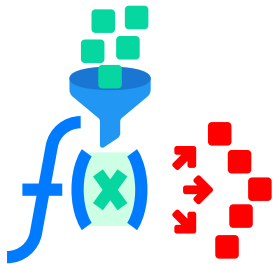


Untyped Functions

Many-Sorted Logic

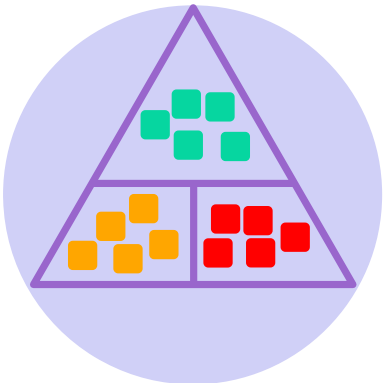


Typed Domain

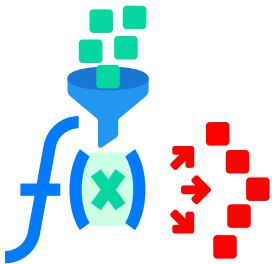


Typed Functions

Order-Sorted Logic

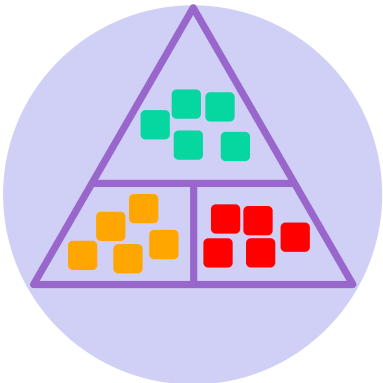


Order-Sorted Domain

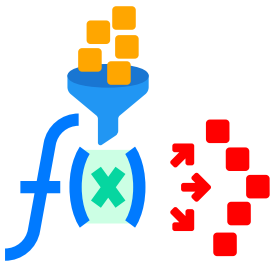


Typed Functions

Order-Sorted Logic



Order-Sorted Domain



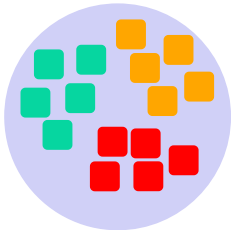
Typed Functions

Sorted Logic

Sorted first-order logic

Partial function logic

Sorted Logic

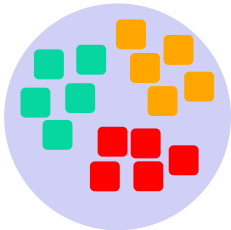


Sorted first-order logic

There are *types*

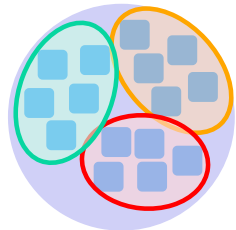
Partial function logic

Sorted Logic



Sorted first-order logic

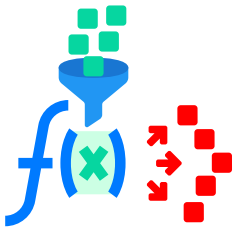
There are *types*



Partial function logic

Unary total predicates

Sorted Logic



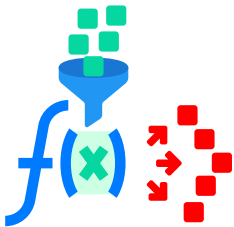
Sorted first-order logic

There are *types*
Predicates and functions are typed

Partial function logic

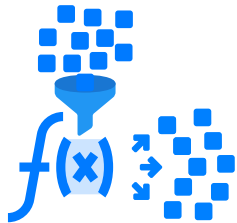
Unary total predicates

Sorted Logic



Sorted first-order logic

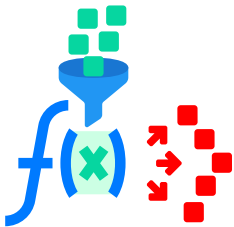
There are *types*
Predicates and functions are typed



Partial function logic

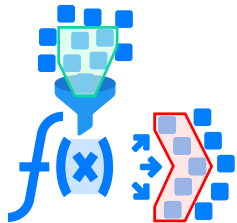
Unary total predicates
Predicates and functions are partial

Sorted Logic



Sorted first-order logic

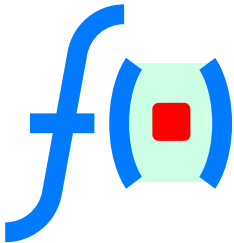
There are *types*
Predicates and functions are typed



Partial function logic

Unary total predicates
Predicates and functions are partial

Sorted Logic



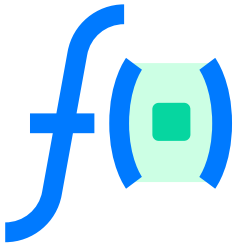
Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed

Partial function logic

Unary total predicates
Predicates and functions are partial

Sorted Logic



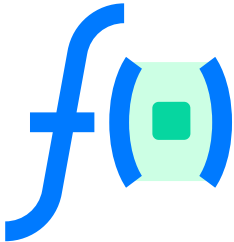
Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed

Partial function logic

Unary total predicates
Predicates and functions are partial

Sorted Logic



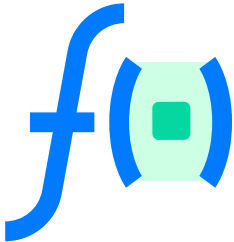
Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed

Partial function logic

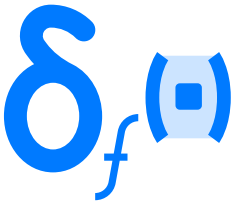
Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded

Sorted Logic



Sorted first-order logic

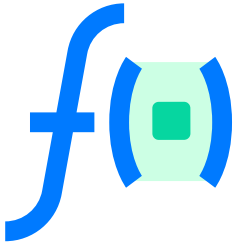
There are *types*
Predicates and functions are typed
Expressions have to be well-typed



Partial function logic

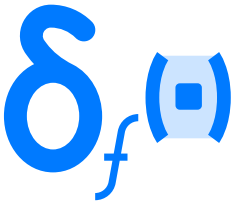
Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded

Sorted Logic



Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes



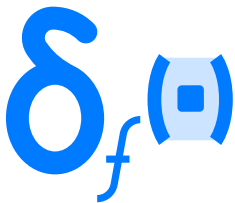
Partial function logic

Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded

Sorted Logic

Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes



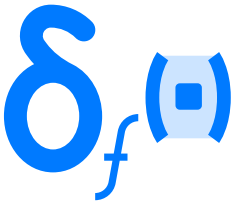
Partial function logic

Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded
This eliminates undefinedness

Sorted Logic

Sorted first-order logic

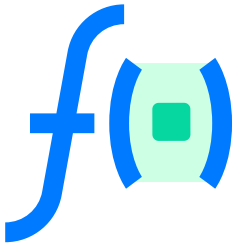
There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes
This is decidable in polynomial time



Partial function logic

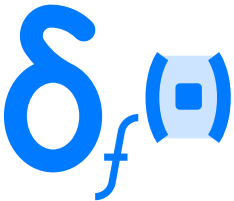
Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded
This eliminates undefinedness

Sorted Logic



Sorted first-order logic

There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes
This is decidable in polynomial time



Partial function logic

Unary total predicates
Predicates and functions are partial
Expressions have to be well-guarded
This eliminates undefinedness
This is decidable in polynomial time

Order-sorted logic vs Partial function logic

Order-sorted logic $\leq$ Partial function logic

Can Order-sorted logic be reduced to the Logic of Partial Functions?

Order-sorted logic $\leq$ Partial function logic

Can Order-sorted logic be reduced to the Logic of Partial Functions?

Can Well-typedness be restored via Well-guardedness?

Order-sorted logic $\leq$ Partial function logic

Can Order-sorted logic be reduced to the Logic of Partial Functions?

Can Well-typedness be restored via Well-guardedness?

Is there a correspondence between the models?

Sorted Logic

Order-sorted logic $\leq$ Partial function logic

Can Order-sorted logic be reduced to the Logic of Partial Functions?

Can Well-typedness be restored via Well-guardedness?

Is there a correspondence between the models?

Can Inductive Data-types be expressed with Inductive Definitions?

Reduction to the partial function logic

$$\textit{type Country}$$
$$\textit{type Human}$$
$$\textit{KingOf} : \textit{Country} \to \textit{Human}$$
$$\textit{Charles} : \textit{Human}$$

$$\exists c[\textit{Country}] : \textit{KingOf}(c) = \textit{Charles}.$$

Reduction to the partial function logic

$$\textit{type Country}$$
$$\textit{type Human}$$
$$\textit{KingOf} : \textit{Country} \to \textit{Human}$$
$$\textit{Charles} : \textit{Human}$$

$$\exists c[\textit{Country}] : \textit{KingOf}(c) = \textit{Charles}.$$

Reduction to the partial function logic

$$\mathit{type}\ \mathit{Country}$$
$$\mathit{type}\ \mathit{Human}$$
$$\mathit{KingOf} : \mathit{Country} \to \mathit{Human}$$
$$\mathit{Charles} : \mathit{Human}$$

$$\exists c[\mathit{Country}] : \mathit{KingOf}(c) = \mathit{Charles}.$$

Reduction to the partial function logic

$$\textit{type Country}$$
$$\textit{type Human}$$
$$\textit{KingOf} : \textit{Country} \to \textit{Human}$$
$$\textit{Charles} : \textit{Human}$$

$$\exists c[\textit{Country}] : \textit{KingOf}(c) = \textit{Charles}.$$

Reduction to the partial function logic

$$\textit{type Country}$$
$$\textit{type Human}$$
$$\textit{KingOf} : \textit{Country} \to \textit{Human}$$
$$\textit{Charles} : \textit{Human}$$

$$\exists c[\textit{Country}] : \textit{KingOf}(c) = \textit{Charles}.$$

Reduction to the partial function logic

$$\mathit{type}\ \mathit{Country}$$
$$\mathit{type}\ \mathit{Human}$$
$$\mathit{KingOf} : \mathit{Country} \to \mathit{Human}$$
$$\mathit{Charles} : \mathit{Human}$$

$$\exists c[\mathit{Country}] : \mathit{KingOf}(c) = \mathit{Charles}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$
$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$
$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1 : & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0 : & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rr} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rr} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rr} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

$$\begin{array}{rl} \mathit{Country}/1 & \delta_{\mathit{Country}}/1 \\ \mathit{Human}/1 & \delta_{\mathit{Human}}/1 \\ \mathit{KingOf}/1: & \delta_{\mathit{KingOf}}/1 \\ \mathit{Charles}/0: & \delta_{\mathit{Charles}}/0 \end{array}$$

$$\forall x : \delta_{\mathit{Country}}(x). \qquad \forall x : \delta_{\mathit{Human}}(x).$$

$$\forall x : \delta_{\mathit{KingOf}}(x) \Leftrightarrow \mathit{Country}(x). \qquad \forall x : \delta_{\mathit{KingOf}}(x) \Rightarrow \mathit{Human}(\mathit{KingOf}(x)).$$

$$\delta_{\mathit{Charles}}(). \qquad \delta_{\mathit{Charles}}() \Rightarrow \mathit{Human}(\mathit{Charles}).$$

$$\exists c : \textbf{if } \mathit{Country}(c) \textbf{ then } \mathit{KingOf}(c) = \mathit{Charles} \textbf{ else } \mathit{false} \textbf{ fi}.$$

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
type T	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** *false* **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
type T	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** *false* **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** $false$ **fi**.

Reduction to the partial function logic

OS Voc	PF Voc	PF Theory
$type\ T$	$T/1 \quad \delta_T/1$	$\forall x : \delta_T(x). \qquad \exists x : T(x).$
$T \prec S$		$\forall x : T(x) \Rightarrow S(x).$
$T_1, T_2 \prec S$		$\neg\exists x : T_1(x) \wedge T_2(x).$
$p : T_1, \ldots, T_n \to B$	$p/n \quad \delta_p/n$	$\forall x_1, \ldots, x_n : \delta_p(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
$f : T_1, \ldots, T_n \to T$	$f/n : \quad \delta_f/n$	$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Leftrightarrow T_1(x_1) \wedge \cdots \wedge T_n(x_n).$
		$\forall x_1, \ldots, x_n : \delta_f(x_1, \ldots, x_n) \Rightarrow T(f(x_1, \ldots, x_n)).$
$\exists x[T] : \phi.$		$\exists x :$ **if** $T(x)$ **then** ϕ **else** *false* **fi**.

Reduction to the partial function logic - Results

Theorem 1. Given an FO(OS) theory T over vocabulary Σ and their translation Σ' and T' to FO(PF), then the FO(OS) theory T is well-typed iff the FO(PF) theory T' is well-guarded.

Reduction to the partial function logic - Results

Theorem 1. Given an FO(OS) theory T over vocabulary Σ and their translation Σ' and T' to FO(PF), then the FO(OS) theory T is well-typed iff the FO(PF) theory T' is well-guarded.

Theorem 2. Given an FO(OS) theory T and its FO(PF) translation T', the following holds:

$$\mathcal{M}(T) \cong \mathcal{M}(T')$$

Inductive data-types

→ **Conclusion**

Inductive data-types

Type $\mathbb{N}\mathrm{at}$ where

$z : \mathbb{N}\mathrm{at}$

$s : \mathbb{N}\mathrm{at} \to \mathbb{N}\mathrm{at}$

Inductive data-types

Type $\mathbb{N}\mathrm{at}$ where

$z : \mathbb{N}\mathrm{at}$

$s : \mathbb{N}\mathrm{at} \to \mathbb{N}\mathrm{at}$

Inductive data-types

Type $\mathbb{N}\mathrm{at}$ where

$z : \mathbb{N}\mathrm{at}$

$s : \mathbb{N}\mathrm{at} \to \mathbb{N}\mathrm{at}$

Inductive data-types

Type $\mathbb{N}\mathrm{at}$ where

$z : \mathbb{N}\mathrm{at}$

$s : \mathbb{N}\mathrm{at} \to \mathbb{N}\mathrm{at}$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$N/1 \qquad z/0 : \qquad s/1 :$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$N/1 \qquad z/0: \qquad s/1:$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$$N/1 \qquad z/0: \qquad s/1:$$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$N/1 \qquad z/0 : \qquad s/1 :$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Inductive data-types

$N/1 \qquad z/0: \qquad s/1:$

$$\forall x : \delta_N(x).$$

$$\delta_z. \qquad \delta_z \Rightarrow N(z).$$

$$\forall x : \delta_s(x) \Leftrightarrow N(x). \qquad \forall x : \delta_s(x) \Rightarrow N(s(x)).$$

$$\left\{ \begin{array}{l} N(z). \\ \forall n : N(s(n)) \leftarrow N(n). \end{array} \right\}$$

$$\forall x : \delta_s(x) \Rightarrow s(x) \neq z. \qquad \forall x, y : \delta_s(x) \wedge \delta_s(y) \wedge s(x) = s(y) \Rightarrow x = y.$$

Conclusion

Conclusion

Partial Functions

Sorted Logic

Conclusion

Partial Functions

Undefinedness

Sorted Logic

Conclusion

Partial Functions

Undefinedness

Sorted Logic

Category clashes

Conclusion

Partial Functions

Undefinedness

Well-guarded

Sorted Logic

Category clashes

Conclusion

Partial Functions

Undefinedness

Well-guarded

Sorted Logic

Category clashes

Well-typed

Conclusion

Partial Functions

Undefinedness
Well-guarded
Recursive definitions

Sorted Logic

Category clashes
Well-typed

Conclusion

Partial Functions

Undefinedness
Well-guarded
Recursive definitions

Sorted Logic

Category clashes
Well-typed
Inductive data-types

Conclusion

Partial Functions

Undefinedness
Well-guarded
Recursive definitions

Sorted Logic

Category clashes
Well-typed
Inductive data-types

Reduction

Conclusion

Partial Functions

Undefinedness
Well-guarded
Recursive definitions

Sorted Logic

Category clashes
Well-typed
Inductive data-types

Reduction
Well-typedness via Well-Guardedness

Conclusion

Partial Functions

Undefinedness
Well-guarded
Recursive definitions

Sorted Logic

Category clashes
Well-typed
Inductive data-types

Reduction
Well-typedness via Well-Guardedness
Model correspondence

Thank you for your attention!

Questions?

What Is More Fundamental: Partial Functions or Types?

Đorđe Marković

dorde.markovic@aau.at