Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Đorđe Marković

12. 12. 2025 – University of Klagenfurt

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Đorđe Marković

12. 12. 2025 – University of Klagenfurt

Slides reused from my public PhD defense

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Đorđe Marković

12. 12. 2025 – University of Klagenfurt

Slides reused from my public PhD defense
PhD Supervisor Prof. dr. Marc Denecker

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Đorđe Marković

12. 12. 2025 – University of Klagenfurt

Slides reused from my public PhD defense
PhD Supervisor Prof. dr. Marc Denecker

Recently joined the EX3 project with Dr. Alice Tarzariol

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Knowledge Representation with Partial Functions and Intensional Concepts

Theory and Applications

Knowledge Representation with Partial Functions and Intensional Concepts

Decision Modeling

Knowledge Representation with Partial Functions and Intensional Concepts

~~Decision Modeling~~

Knowledge Representation

John McCarthy

“As soon as it works, no one calls it AI any more.”

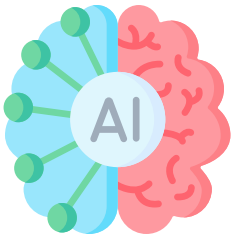
As cited by Bertrand Meyer

Artificial intelligence – Major goals

Artificial general intelligence

Computer vision

General game playing



Recursive self-improvement

Planning, Robotics

Knowledge representation

Natural language processing

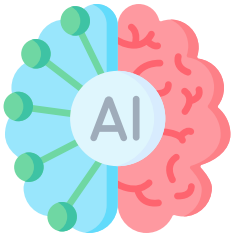
Intelligent agent, AI safety

Artificial intelligence – Approaches

Machine learning

Deep learning

Systems integration



Evolutionary algorithms

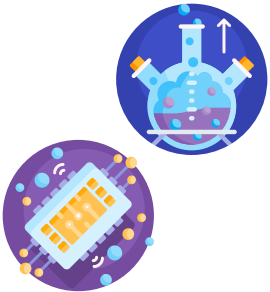
Bayesian networks

Symbolic

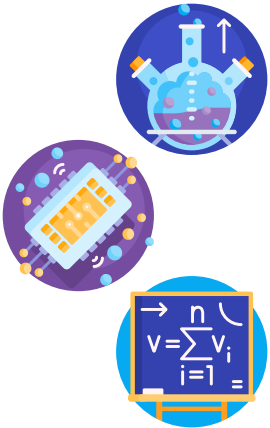
Hybrid intelligent systems

Knowledge representation

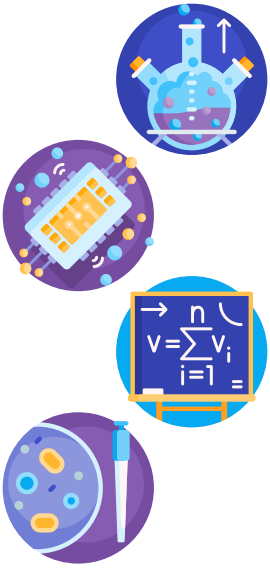
Knowledge representation



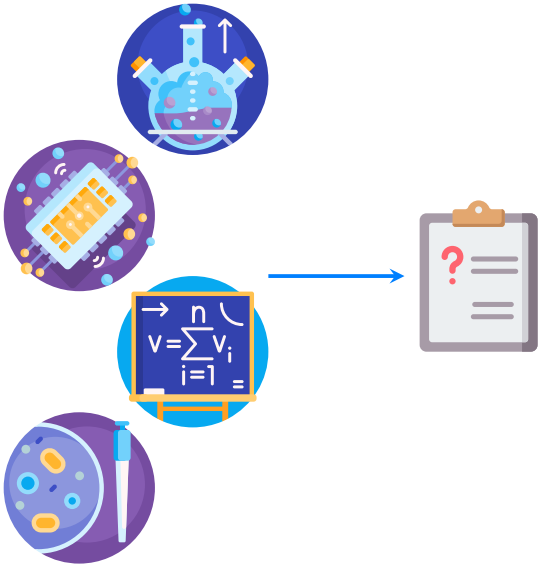
Knowledge representation



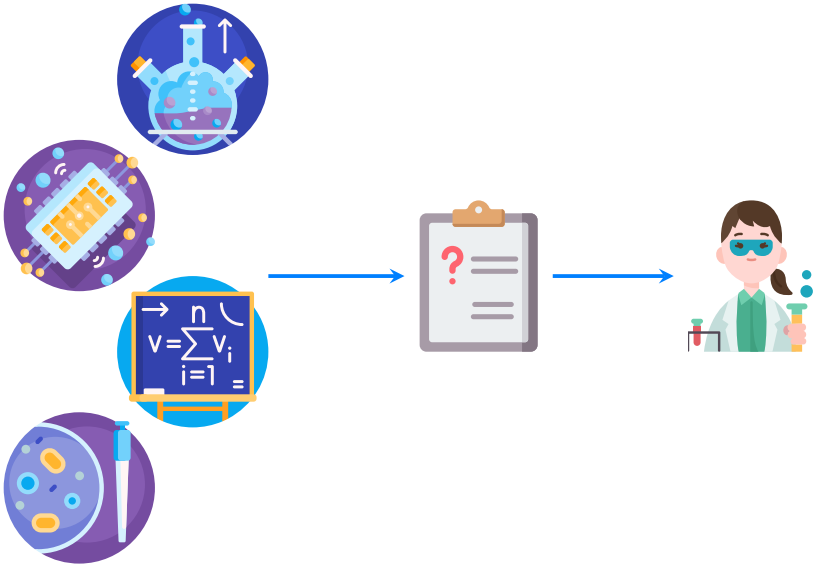
Knowledge representation



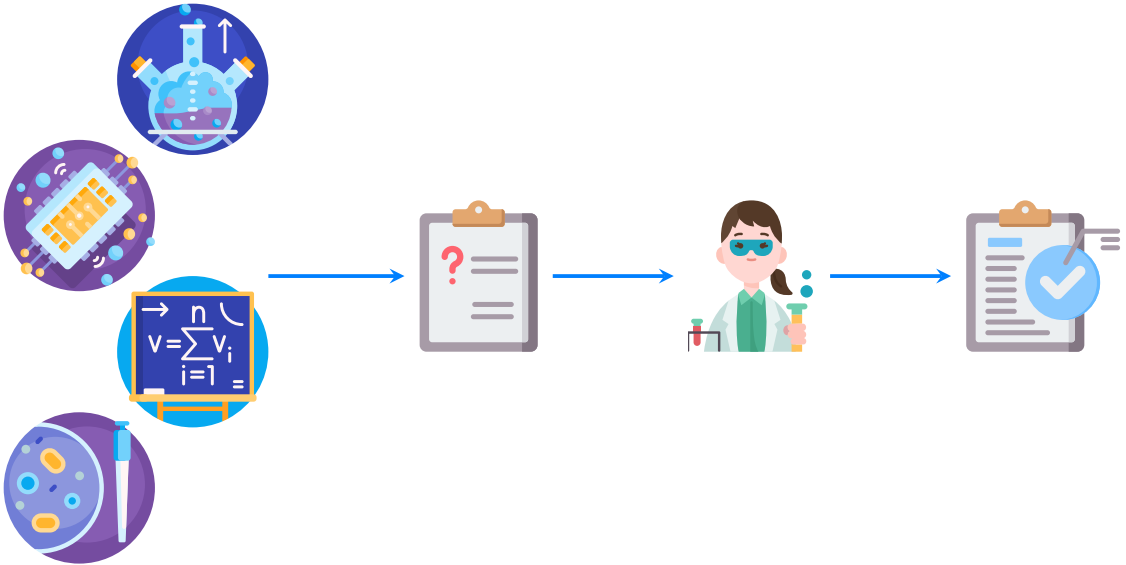
Knowledge representation



Knowledge representation

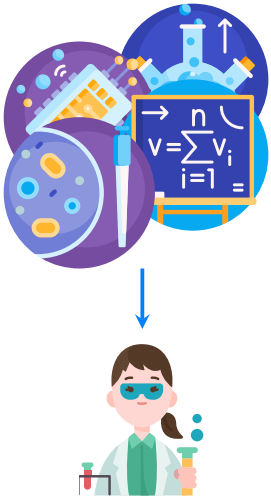


Knowledge representation

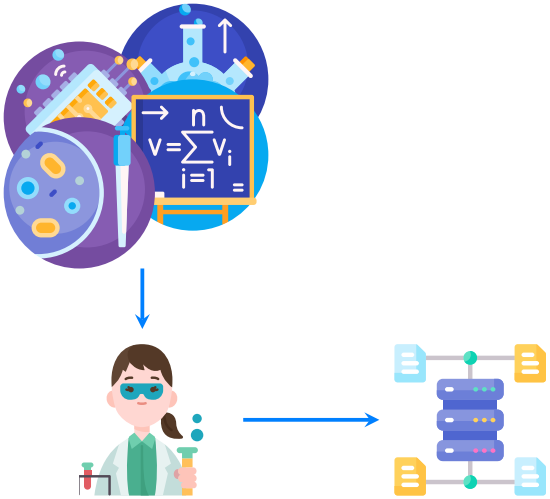


Knowledge representation

Knowledge representation



Knowledge representation



Knowledge representation

Knowledge representation

Knowledge representation

‘World Models,’ an Old Idea in AI, Mount a Comeback

Quantamagazine (2025)

Knowledge representation language

Knowledge representation language

Natural language?

Knowledge representation language – Natural language?

Smoking is not allowed in all parts of the plane!

Knowledge representation language – Natural language?

In all parts of the plane, it is not true that smoking is allowed!

Knowledge representation language – Natural language?

In all parts of the plane, it is not true that smoking is allowed!

It is not true that smoking is allowed in all parts of the plane!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Knowledge representation language – Logic!

Smoking is not allowed in all parts of the plane!

Knowledge representation language – Logic!

Smoking is not allowed in all parts of the plane!

$\forall x$:

Knowledge representation language – Logic!

Smoking is not allowed in all parts of the plane!

$\forall x : \neg$

Knowledge representation language – Logic!

Smoking is not allowed in all parts of the plane!

$$\forall x : \neg smoking\_allowed(x)$$

Knowledge representation language – Logic!

Smoking is not allowed in all parts of the plane!

$\forall x : \neg smoking\_allowed(x)$

$\neg \forall x : smoking\_allowed(x)$

Knowledge representation language – Logic!

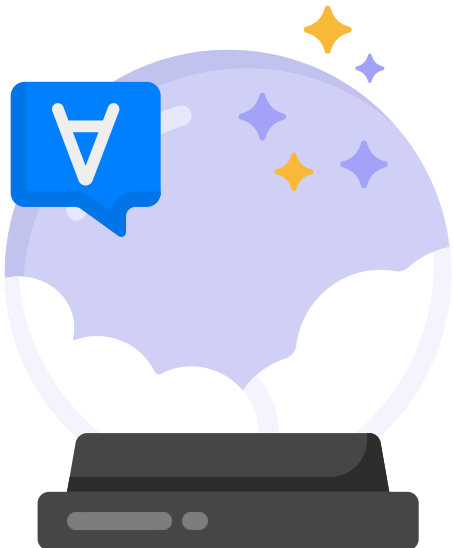
Smoking is not allowed in all parts of the plane!

$$\forall x : \neg smoking\_allowed(x)$$

$$\neg \forall x : smoking\_allowed(x)$$

Knowledge representation language – Logic

Knowledge representation language – Logic



Partial Functions

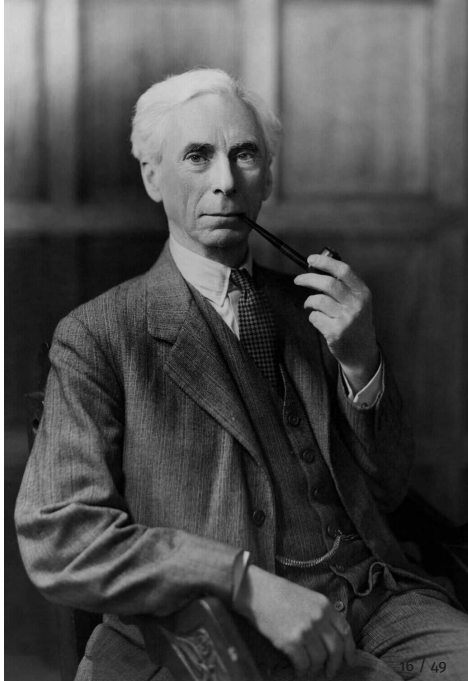
Total Functions!

Partial Functions!

Bertrand Russell

"By the law of excluded middle, either "A is B" or "A is not B" must be true. Hence either "the present King of France is bald" or "the present King of France is not bald" must be true. Yet if we enumerated the things that are bald, and then the things that are not bald, we should not find the present King of France in either list. Hegelians, who love a synthesis, will probably conclude that he wears a wig."

On denoting, Mind (1905).

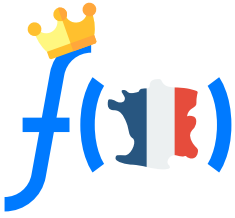


Partial Functions

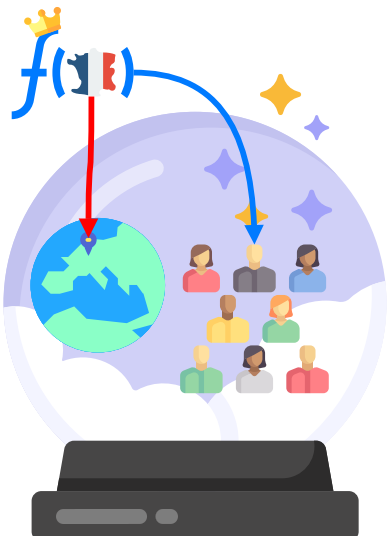
The present King of France is bald.

Partial Functions

The present King of France is bald.

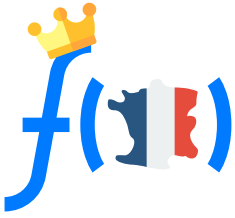


Partial Functions



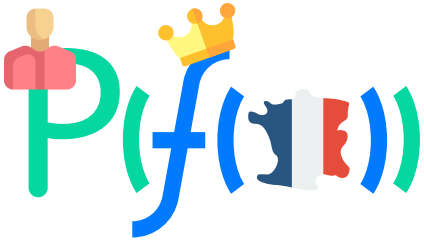
Partial Functions

The present King of France is bald.



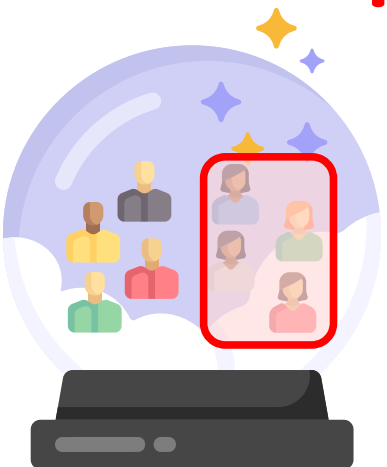
Partial Functions

The present King of France is bald.



Partial Functions

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.

The wife of the king of France is the queen of France.

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.

The wife of the king of France is the queen of France.

For all x and y, if x is the king of France, then if y is the wife of x then y is the queen of France.

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.

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

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

Peter Strawson

Such a sentence does not have a truth value!

Partial Functions in Knowledge Representation

The present King of France is bald.

Partial Functions in Knowledge Representation

The present King of France is bald.

Such a sentence is ambiguous!

Partial Functions in Knowledge Representation

The present King of France is bald.

Such a sentence is ambiguous!

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

Partial Functions in Knowledge Representation

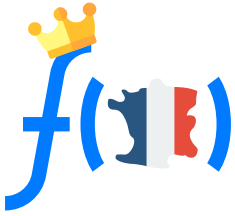
The present King of France is bald.

Such a sentence is ambiguous!

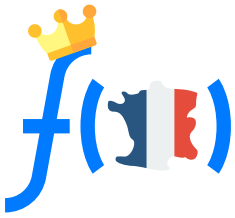
There exists an x which is the king of France and x is bald.

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

Partial Functions in Knowledge Representation



Partial Functions in Knowledge Representation



KoF

Partial Functions in Knowledge Representation

Partial Functions in Knowledge Representation

$Bald(KoF)$

The present king of France exists.

The present king of France exists.

$$\delta_{KoF}$$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

if δ_{KoF} **then** $Bald(KoF)$ **else** $false$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

if δ_{KoF} **then** $Bald(KoF)$ **else** $false$

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

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

if δ_{KoF} **then** $Bald(KoF)$ **else** $false$

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

if δ_{KoF} **then** $Bald(KoF)$ **else** $true$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

$$\delta_{KoF} \overrightarrow{\wedge} Bald(KoF)$$

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

if δ_{KoF} **then** $Bald(KoF)$ **else** $true$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

$$\delta_{KoF} \overrightarrow{\wedge} Bald(KoF)$$

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

$$\delta_{KoF} \Rightarrow Bald(KoF)$$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

$$[[Bald(KoF)]]$$

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

$$\delta_{KoF} \rightrightarrows Bald(KoF)$$

Partial Functions in Knowledge Representation

The present King of France exists and he is bald.

$$[[Bald(KoF)]]$$

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

$$\langle\langle Bald(KoF)\rangle\rangle$$

Partial Functions in Knowledge Representation

$$f(x)$$

Partial Functions in Knowledge Representation

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

Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

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

Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

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

Well-guarded formula

Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

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

Non-ambiguous formula

Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

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

Polynomially Decidable

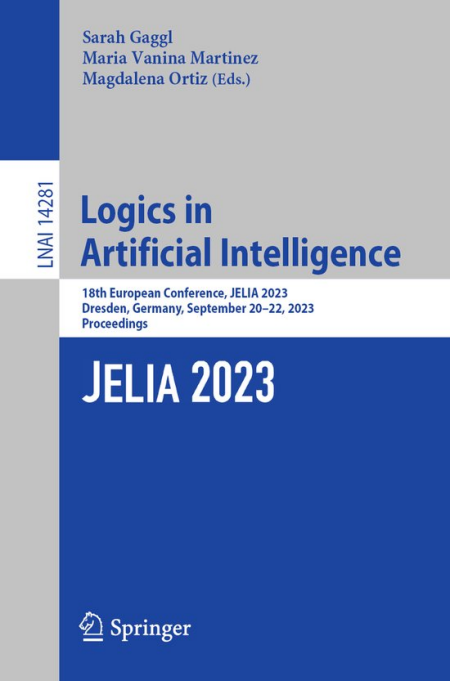
Partial Functions in Knowledge Representation

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

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

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

Well-defined formula



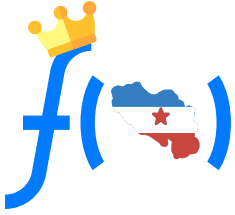
Markovic D., Bruynooghe M., Denecker M.,

Towards Systematic Treatment of Partial Functions in Knowledge Representation

European Conference on Logics in Artificial Intelligence (2023)

Partial function logic

Guarding Partial Functions



Guarding Partial Functions

$$KingOf(YU)$$

Guarding Partial Functions

Guarding Partial Functions

$$\Delta(KingOf(YU))$$

Guarding Partial Functions

$$\Delta(KingOf(YU))$$

Term guarding

Guarding Partial Functions

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

Guarding Partial Functions

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

$$\delta_{KoF}.$$

Guarding Partial Functions

$$\textbf{if } \delta_f(x) \textbf{ then } \ldots f(x) \ldots \textbf{ else } \chi$$

$$\delta_{KoF}. \qquad Bald(KoF).$$

Guarding Partial Functions

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

$\delta_{KoF}.$ $\quad$ $Bald(KoF).$

Indirect guarding

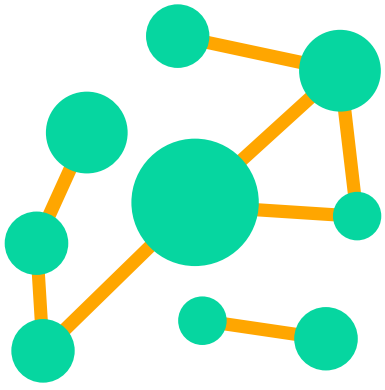
Defining Partial Functions

Defining Partial Functions

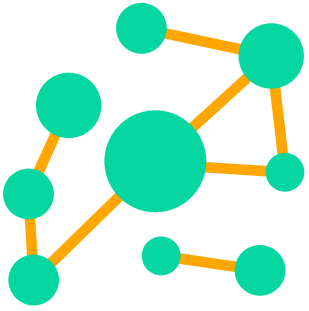
Extending classical logic with inductive definitions.

Denecker M., International Conference on Computational Logic (2000).

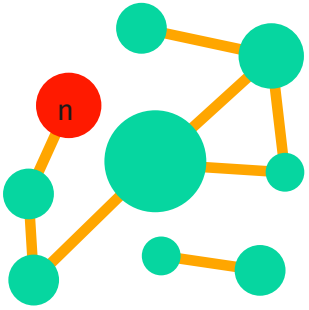
Defining Partial Functions



Defining Partial Functions

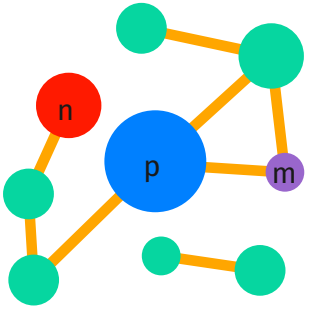


Defining Partial Functions



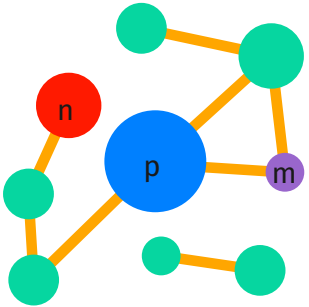
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \\ \\ \end{array} \right\}$$

Defining Partial Functions



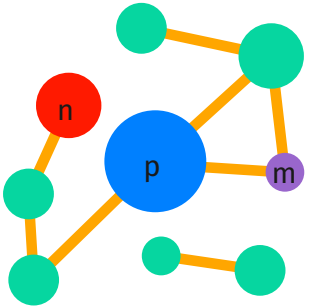
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = \end{array} \right\}$$

Defining Partial Functions



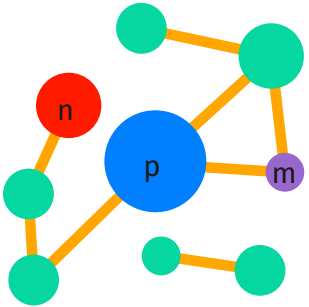
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \end{array} \right\}$$

Defining Partial Functions



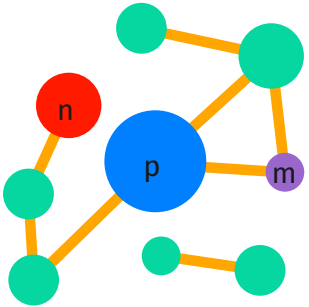
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \end{array} \right\}$$

Defining Partial Functions



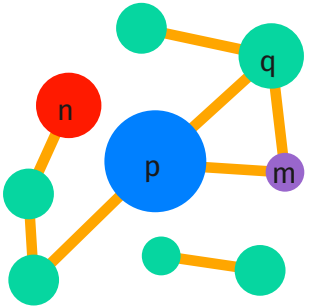
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \wedge (n \neq m) \end{array} \right\}$$

Defining Partial Functions



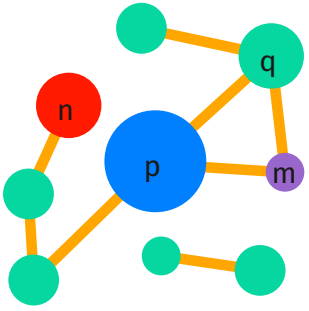
$$\left\{ \begin{array}{l} \forall n : dist(n, n) = 0 \\ \forall n, m, p : dist(n, m) = dist(n, p) + 1 \leftarrow \Delta(dist(n, p) + 1) \wedge (n \neq m) \wedge E(p, m) \end{array} \right\}$$

Defining Partial Functions



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

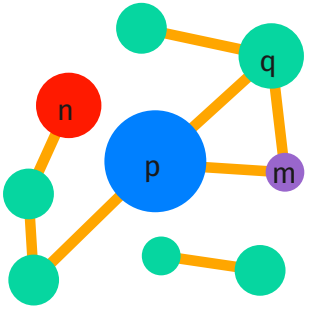
Defining Partial Functions



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

Recursive definitions

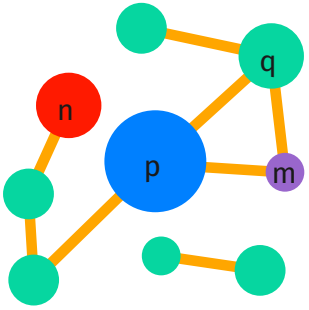
Defining Partial Functions



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

Recursive definitions

Defining Partial Functions



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

Recursive definitions

Term and Indirect Guarding

Term and Indirect Guarding

Recursive Definitions

Term and Indirect Guarding

Recursive Definitions

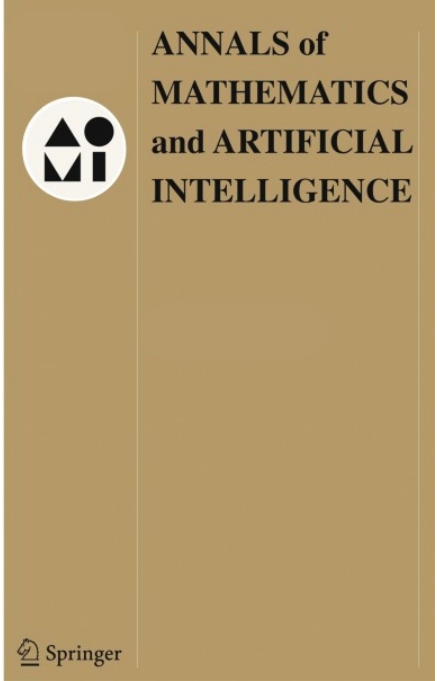
Reduction to semantics of inductive definitions of sets

Term and Indirect Guarding

Recursive Definitions

Reduction to semantics of inductive definitions of sets

Correspondence with functional programming



Markovic D., Van den Eede R., Denecker M.

A Prudent Logic of Partial Functions

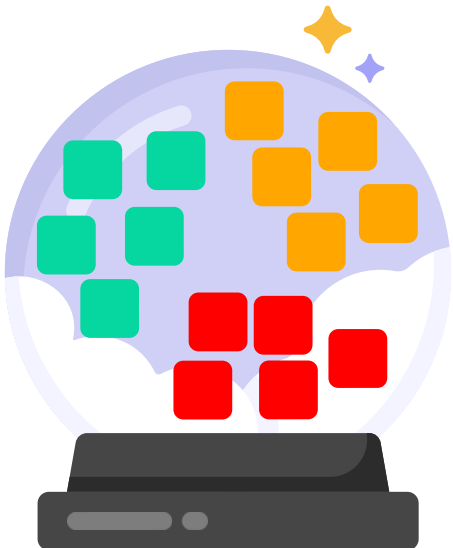
Accepted for publication in Annals of Mathematics and Artificial Intelligence

A Prudent Logic of Partial Functions

Sorted Logic
with Intensional Concepts

Unsorted logic

Sorted Logic

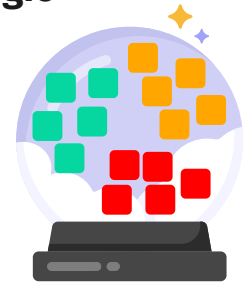


Sorted Logic

Sorted first-order logic

Partial function logic

Sorted Logic

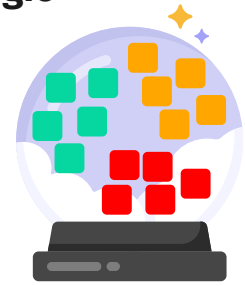


Sorted first-order logic

There are *types*

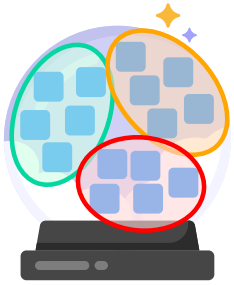
Partial function logic

Sorted Logic



Sorted first-order logic

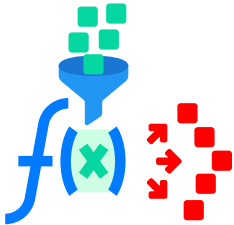
There are *types*



Partial function logic

Unary total predicates

Sorted Logic



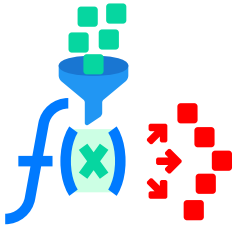
Sorted first-order logic

There are *types*
Predicates and functions are typed

Partial function logic

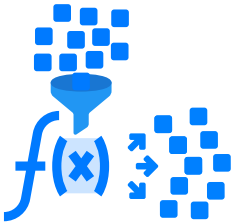
Unary total predicates

Sorted Logic



Sorted first-order logic

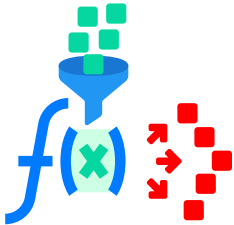
There are *types*
Predicates and functions are typed



Partial function logic

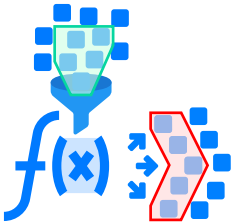
Unary total predicates
Partial predicates and functions

Sorted Logic



Sorted first-order logic

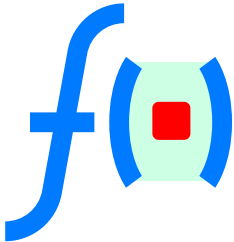
There are *types*
Predicates and functions are typed



Partial function logic

Unary total predicates
Partial predicates and functions

Sorted Logic



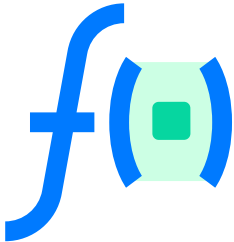
Sorted first-order logic

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

Partial function logic

Unary total predicates
Partial predicates and functions

Sorted Logic



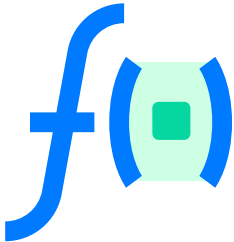
Sorted first-order logic

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

Partial function logic

Unary total predicates
Partial predicates and functions

Sorted Logic



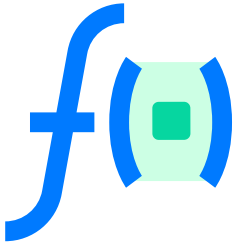
Sorted first-order logic

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

Partial function logic

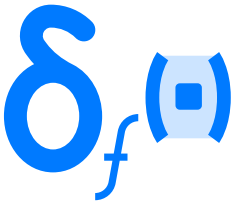
Unary total predicates
Partial predicates and functions
Expressions have to be well-guarded

Sorted Logic



Sorted first-order logic

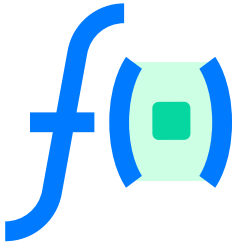
There are *types*
Predicates and functions are typed
Expressions have to be well-typed



Partial function logic

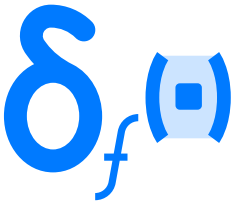
Unary total predicates
Partial predicates and functions
Expressions have to be well-guarded

Sorted Logic



Sorted first-order logic

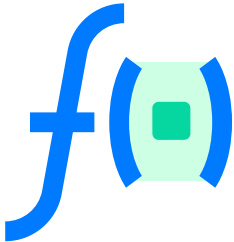
There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes



Partial function logic

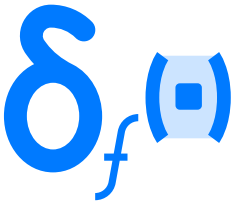
Unary total predicates
Partial predicates and functions
Expressions have to be well-guarded

Sorted Logic



Sorted first-order logic

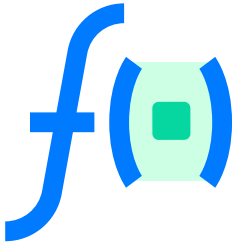
There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes



Partial function logic

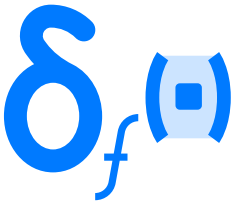
Unary total predicates
Partial predicates and functions
Expressions have to be well-guarded
This eliminates undefinedness

Sorted Logic



Sorted first-order logic

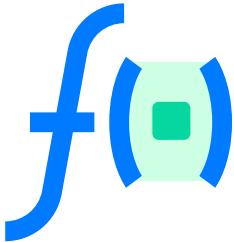
There are *types*
Predicates and functions are typed
Expressions have to be well-typed
This eliminates category clashes
This is decidable in polynomial time



Partial function logic

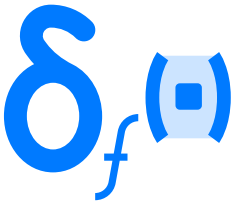
Unary total predicates
Partial predicates and functions
Expressions have to be well-guarded
This eliminates undefinedness

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
Partial predicates and functions
Expressions have to be well-guarded
This eliminates undefinedness
This is decidable in polynomial time

Sorted first-order logic vs Partial function logic

Sorted Logic

Sorted first-order logic $\leq$ Partial function logic

There is a reduction

Sorted Logic

Sorted first-order logic $\leq$ Partial function logic

There is a reduction

Well-typedness can be restored via well-guardedness

Sorted Logic

Sorted first-order logic $\leq$ **Partial function logic**

There is a reduction

Well-typedness can be restored via well-guardedness

Many-sorted logic, **Order-sorted logic**, and extensions

Sorted Logic

Sorted first-order logic $\leq$ Partial function logic

There is a reduction

Well-typedness can be restored via well-guardedness

Many-sorted logic, **Order-sorted logic**, and extensions

Instead of extending typed logic with partial functions, we can do the reverse

Sorted Logic

Sorted first-order logic $\leq$ Partial function logic

There is a reduction

Well-typedness can be restored via well-guardedness

Many-sorted logic, **Order-sorted logic**, and extensions

Instead of extending typed logic with partial functions, we can do the reverse

Using inductive definitions we can define **inductive data-types**

Reduction to the partial function logic

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

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

Reduction to the partial function logic

$$KingOf : Country \to Human$$
$$Charles : Human$$

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

Reduction to the partial function logic

$$KingOf : Country \to Human$$
$$Charles : Human$$

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

Reduction to the partial function logic

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

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

Reduction to the partial function logic

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

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

Reduction to the partial function logic

$$KingOf : Country \to Human$$
$$Charles : Human$$

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

Reduction to the partial function logic

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

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

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

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

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

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

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

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

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

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

Markovic D., Denecker M.

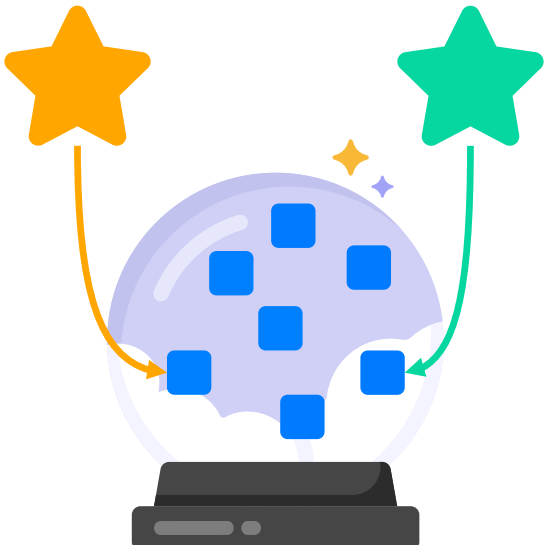
Types and Partial Functions

To be submitted

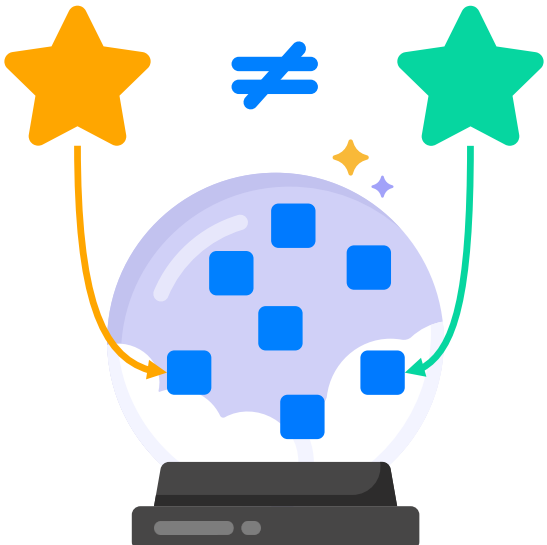
Sorted Logic
with Intensional Concepts

Intensional Logic

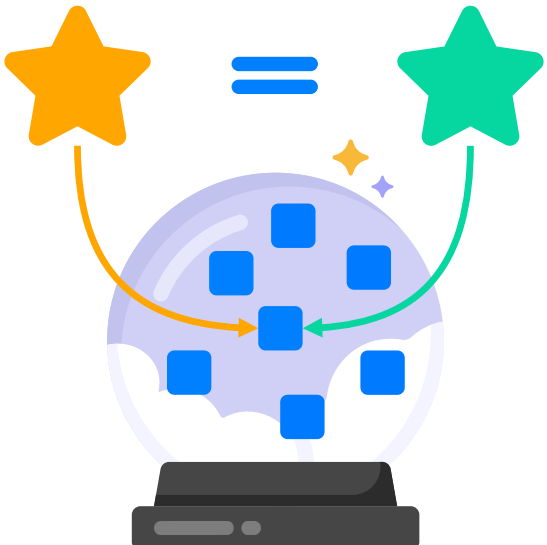
Intensional Logic



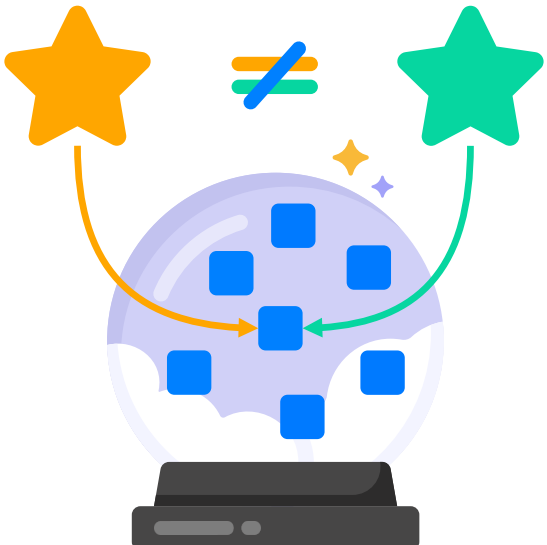
Intensional Logic



Intensional Logic



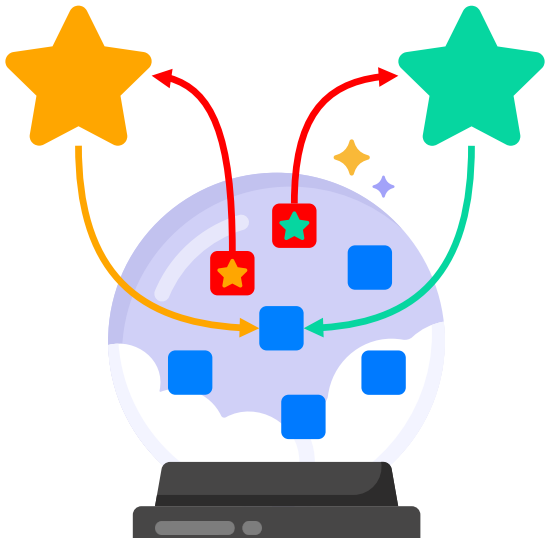
Intensional Logic



Intensional Logic

Quantification and Aggregation over Concepts of the Ontology.

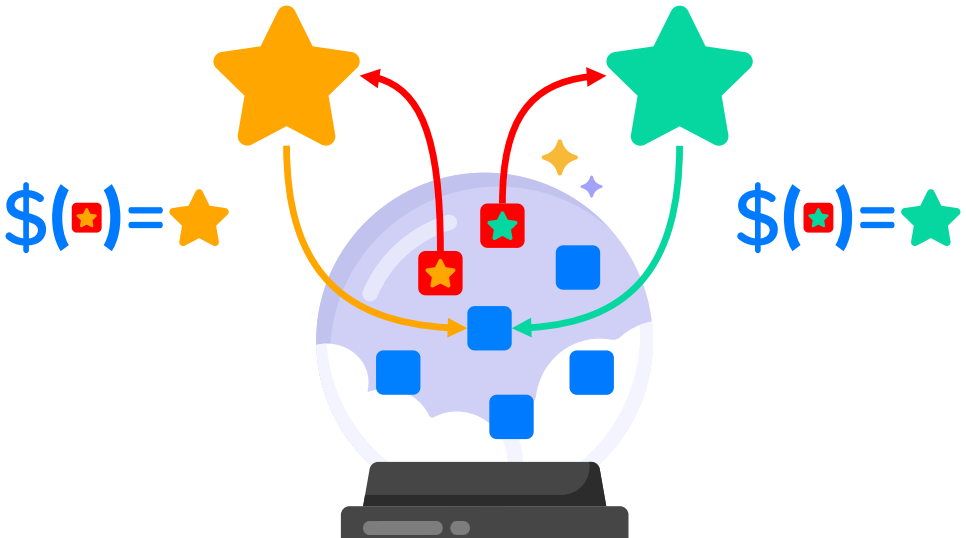
Carbonnelle P., Van der Hallen M., Denecker M., ICLP (2023).



Intensional Logic

Quantification and Aggregation over Concepts of the Ontology.

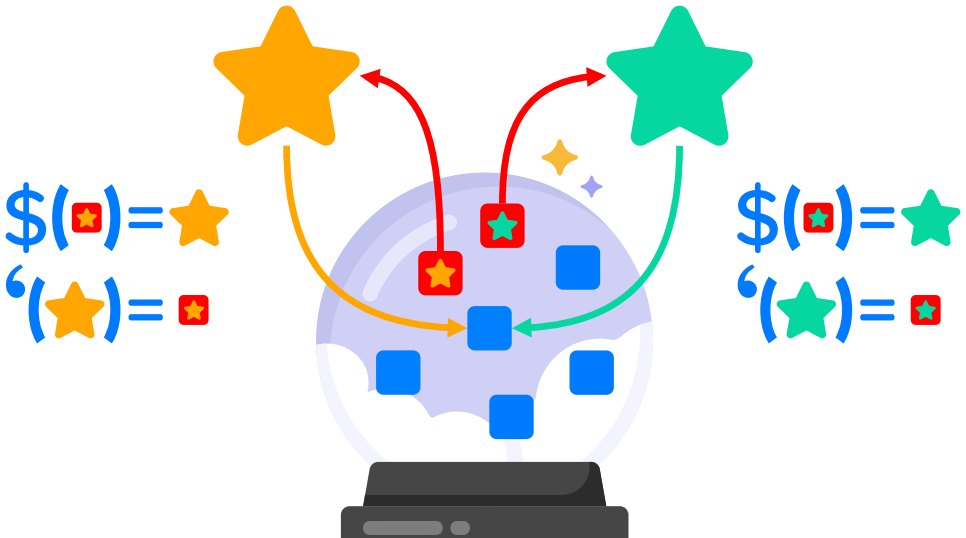
Carbonnelle P., Van der Hallen M., Denecker M., ICLP (2023).



Intensional Logic

Quantification and Aggregation over Concepts of the Ontology.

Carbonnelle P., Van der Hallen M., Denecker M., ICLP (2023).



Intensional Logic

$$\mathit{Old} : \mathit{Human} \to \mathbb{B} \qquad \mathit{Bald} : \mathit{Human} \to \mathbb{B} \qquad \mathit{Tall} : \mathit{Human} \to \mathbb{B}$$

Intensional Logic

$$\mathit{Old} : \mathit{Human} \to \mathbb{B} \qquad \mathit{Bald} : \mathit{Human} \to \mathbb{B} \qquad \mathit{Tall} : \mathit{Human} \to \mathbb{B}$$

$$\mathit{Old}(\mathit{Charles}) \wedge \mathit{Bald}(\mathit{Charles}) \wedge \mathit{Tall}(\mathit{Charles}).$$

Intensional Logic

$$Old : Human \to \mathbb{B} \qquad Bald : Human \to \mathbb{B} \qquad Tall : Human \to \mathbb{B}$$

$$P(`(Old)). \qquad P(`(Bald)). \qquad P(`(Tall))$$

$$Old(Charles) \wedge Bald(Charles) \wedge Tall(Charles).$$

Intensional Logic

$$Old : Human \to \mathbb{B} \qquad Bald : Human \to \mathbb{B} \qquad Tall : Human \to \mathbb{B}$$

$$P(`(Old)). \qquad P(`(Bald)). \qquad P(`(Tall))$$

$$Old(Charles) \wedge Bald(Charles) \wedge Tall(Charles).$$

$$\forall c[\mathbb{C}] : P(c) \Rightarrow \$(c)(Chalrs).$$

Order-sorted Logic with Intensional Concepts

Order-sorted Logic with Intensional Concepts

Order-sorted Logic with Intensional Concepts

$$\forall c[\mathbb{C}\mathrm{at}] \ : \ \mathit{Meow}(c). \qquad \forall d[\mathbb{D}\mathrm{og}] \ : \ \mathit{Bark}(d).$$

Order-sorted Logic with Intensional Concepts

$$\forall a[\mathbb{A}\mathrm{nimal}] : (\mathbb{C}\mathrm{at}(a) \Rightarrow \mathit{Meow}(a)) \wedge ((\mathbb{D}\mathrm{og}(a) \Rightarrow \mathit{Bark}(a)).$$

Order-sorted Logic with Intensional Concepts

$$\forall a[\mathbb{A}\mathrm{nimal}] \;:\; \langle\langle \mathit{Meow}(a) \rangle\rangle \wedge \langle\langle \mathit{Bark}(a) \rangle\rangle.$$

Order-sorted Logic with Intensional Concepts

$$\forall a[\mathbb{A}\mathrm{nimal}] \;:\; \langle\langle \mathit{Meow}(a) \rangle\rangle \wedge \langle\langle \mathit{Bark}(a) \rangle\rangle.$$

$$\mathbb{S}\mathrm{ound} \prec \mathbb{C} := \{\text{`}(\mathit{Meow}), \text{`}(\mathit{Bark})\}$$

Order-sorted Logic with Intensional Concepts

$$\forall a[\mathbb{A}\text{nimal}] \;:\; \forall s[\mathbb{S}\text{ound}] \;:\; \langle\langle \$(s)(a)\rangle\rangle$$

$$\mathbb{S}\text{ound} \prec \mathbb{C} := \{\text{`}(\mathit{Meow}), \text{`}(\mathit{Bark})\}$$

Order-sorted Intensional Logic

Order-sorted logic **Intensional logic**

Order-sorted Intensional Logic

Order-sorted logic **Intensional logic**

Quantification over concepts

Order-sorted Intensional Logic

Order-sorted logic **Intensional logic**

Quantification over concepts

Implicit type guarding $[\![\cdot]\!]$

Order-sorted Intensional Logic

Order-sorted logic **Intensional logic**

Quantification over concepts

Implicit type guarding $[[\cdot]]$

Subtyping Polymorphism

EPTCS 416

Proceedings of the
40th International Conference on
Logic Programming

University of Texas at Dallas, Dallas Texas, USA, October 14-17 2024

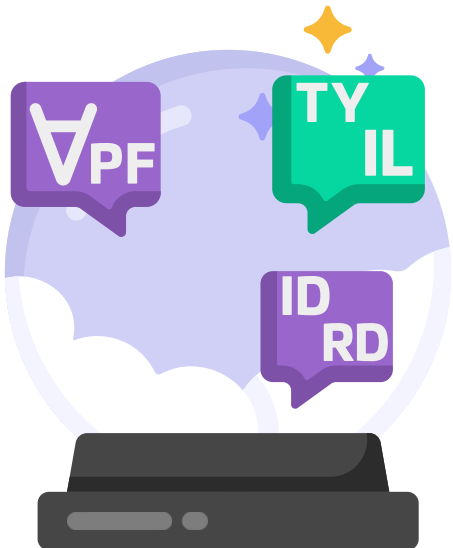
Edited by: Pedro Cabalar, Francesco Fabiano, Martin Gebser, Gopal Gupta and Theresa Swift

Markovic D., Denecker M.,

Order-Sorted Intensional Logic: Expressing Subtyping Polymorphism with Typing Assertions and Quantification over Concepts.

ICLP (2024)

Sorted Logic – with Intensional Concepts

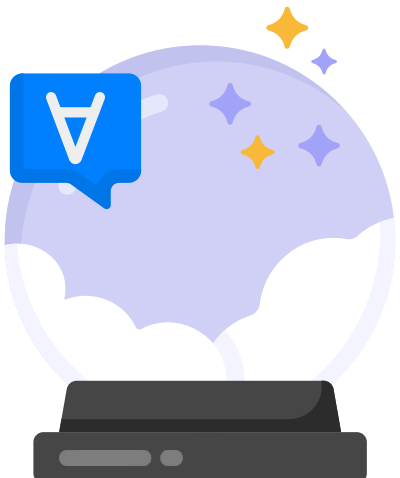


Decision Modeling

~~Decision Modeling~~

Epistemic Decision Modeling and Notation

Conclusion



Partial Functions

Partial Functions

Guards

Partial Functions

Guards

Convenience constructs

Partial Functions

Guards

Convenience constructs

Decidable

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity

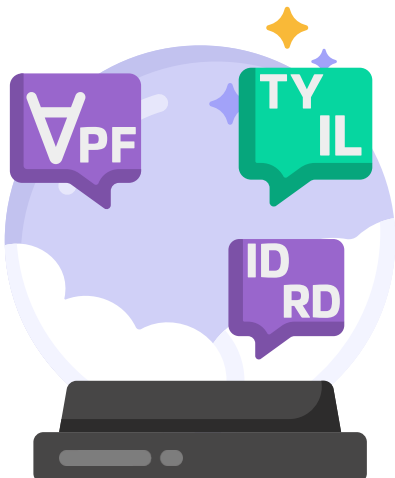
Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

Sorted Logic

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

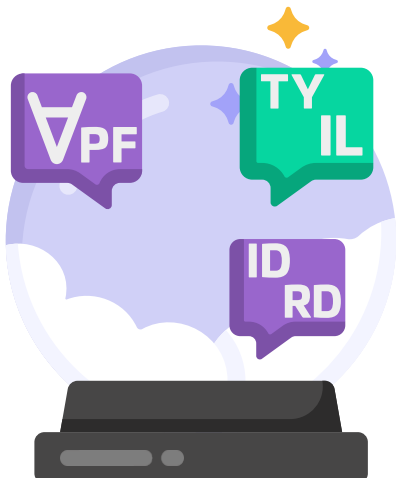


Sorted Logic

Sorted logics

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions



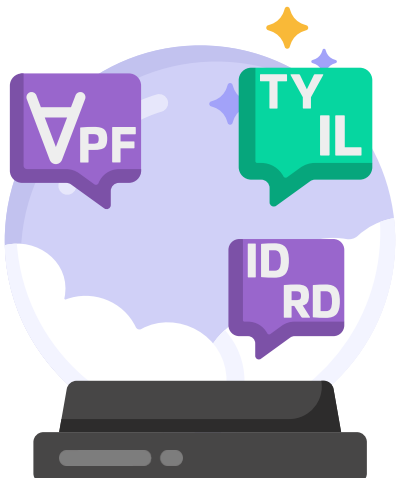
Sorted Logic

Sorted logics

Reduction to PF

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions



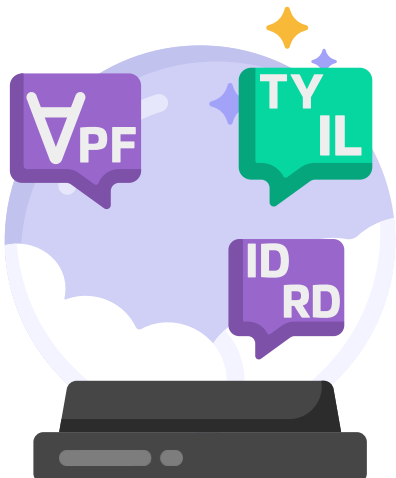
Sorted Logic

Sorted logics
Inductive data-types

Reduction to PF

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions



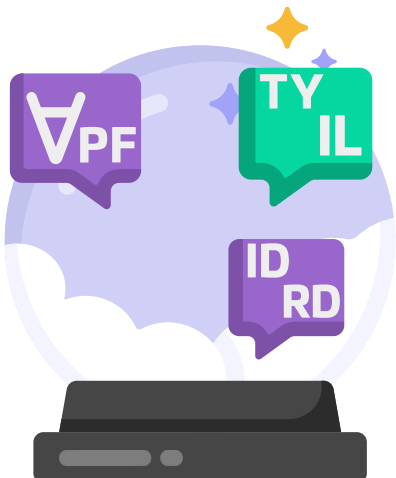
Sorted Logic

Sorted logics
Inductive data-types

Reduction to PF
Intensional logic

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions



Sorted Logic

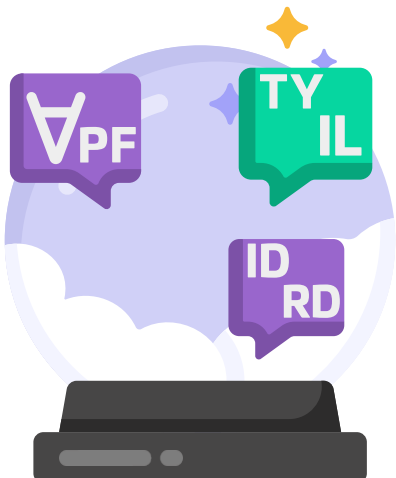
Sorted logics
Inductive data-types

Reduction to PF
Intensional logic

Subtyping polymorphism

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions



Sorted Logic

Sorted logics
Inductive data-types

Reduction to PF
Intensional logic

Subtyping polymorphism

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

Decision Modeling

Sorted Logic

Sorted logics
Inductive data-types
Reduction to PF
Intensional logic
Subtyping polymorphism

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

Decision Modeling

?

Sorted Logic

Sorted logics
Inductive data-types

Reduction to PF
Intensional logic

Subtyping polymorphism

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

Implementation

Decision Modeling

?

Sorted Logic

Sorted logics
Inductive data-types
Dependant types
Reduction to PF
Intensional logic
Subtyping polymorphism

Partial Functions

Guards
Convenience constructs
Decidable
Polinomial complexity
Recursive definitions

Implementation

Decision Modeling

?

Alfred Tarski

“There can be no doubt that the knowledge of logic is of considerable practical importance for everyone who desires to think and to infer correctly.”

Thank you for your attention!

Questions?

Knowledge Representation with Partial Functions and Intensional Concepts

Đorđe Marković

dorde.markovic@aau.at