Language Constructs for Eliminating Ambiguities in Knowledge Representation

Partial Functions, Epistemic Logic, and Order-Sorted Intensional Logic

Đorđe Marković
KU Leuven

October 21, 2024
TTU, Lubbock, Texas, USA

KU LEUVEN

Get the slides

https://djordje.rs/public/presentations/TTU-seminar-2024.pdf

Introduction – KRR Paradigm

Quick poll

How many of you are familiar with/have heard of (raise your hand ✋):

- Knowledge Representation and Reasoning Paradigm?
- Prolog, ASP, Datalog?
- LTL, CTL, ProB, Rodin?
- Using logic for modeling/solving problems?

Quick poll

How many of you are familiar with/have heard of :

- Knowledge Representation and Reasoning Paradigm? 🖐
- Prolog, ASP, Datalog?
- LTL, CTL, ProB, Rodin?
- Using logic for modeling/solving problems?

Quick poll

How many of you are familiar with/have heard of :

- Knowledge Representation and Reasoning Paradigm?
- Prolog, ASP, Datalog? ✋
- LTL, CTL, ProB, Rodin?
- Using logic for modeling/solving problems?

Quick poll

How many of you are familiar with/have heard of :

- Knowledge Representation and Reasoning Paradigm?
- Prolog, ASP, Datalog?
- LTL, CTL, ProB, Rodin? 🖐
- Using logic for modeling/solving problems?

Quick poll

How many of you are familiar with/have heard of :

- Knowledge Representation and Reasoning Paradigm?
- Prolog, ASP, Datalog?
- LTL, CTL, ProB, Rodin?
- Using logic for modeling/solving problems? ✋

Knowledge Representation and Reasoning Paradigm

- Domain specific knowledge
- Different problems/tasks
- Formally model the domain knowledge
- Use reasoning engine for solving different problems with the same formalization

Knowledge Representation and Reasoning Paradigm

- Domain specific knowledge
- Different problems/tasks
- Formally model the domain knowledge
- Use reasoning engine for solving different problems with the same formalization

Knowledge Representation and Reasoning Paradigm

- Domain specific knowledge
- Different problems/tasks
- Formally model the domain knowledge
- Use reasoning engine for solving different problems with the same formalization

Knowledge Representation and Reasoning Paradigm

- Domain specific knowledge
- Different problems/tasks
- Formally model the domain knowledge
- Use reasoning engine for solving different problems with the same formalization

KRR Paradigm - Example

- Domain knowledge: *Graph theory*
- Problems: *Graph coloring, Transitive closure, Reachability, ...*
- Formal language: *First-order logic?*
- Reasoning engine: *SAT/SMT solver?*

KRR Paradigm - Example

- Domain knowledge: *Graph theory*
- Problems: *Graph coloring*, *Transitive closure*, *Reachability*, ...
- Formal language: *First-order logic?*
- Reasoning engine: *SAT/SMT solver?*

KRR Paradigm - Example

- Domain knowledge: *Graph theory*
- Problems: *Graph coloring*, *Transitive closure*, *Reachability*, ...
- Formal language: *First-order logic?*
- Reasoning engine: *SAT/SMT solver?*

KRR Paradigm - Example

- Domain knowledge: *Graph theory*
- Problems: *Graph coloring*, *Transitive closure*, *Reachability*, ...
- Formal language: *First-order logic?*
- Reasoning engine: *SAT/SMT solver?*

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

KRR Paradigm - Example Flow

Example 1:

- We have an FO theory about graph coloring.
- Problem instance: We know the Graph (nodes and edges), and the set of Colors.
- Asking the solver to find a solution (i.e., assignment of colors to the nodes).

Example 2:

- We have an FO theory about graph coloring.
- Problem instance: We know partially the Graph (only nodes), and the set color assignment.
- Asking the solver to find a possible edges of the graph.

Quick poll

How many of you are familiar with (raise your hand 🖐):

- First-order Logic?
- Typed First-order Logic?
- Model semantics?

Quick poll

How many of you are familiar with:

- First-order Logic? ✋
- Typed First-order Logic?
- Model semantics?

Quick poll

How many of you are familiar with:

- First-order Logic?
- Typed First-order Logic? ✋
- Model semantics?

Quick poll

How many of you are familiar with:

- First-order Logic?
- Typed First-order Logic?
- Model semantics? ✋

Intermezzo – First-order Logic

Intermezzo – First-order Logic – Syntax

Term t is defined as:

- Variable: x
- Compound: $f(t_1, \ldots, t_n)$

An FO formula is defined as:

- Atom: $P(t_1, \ldots, t_n)$
- Negation: $\neg\phi$
- Disjunction: $\phi_1 \vee \phi_2$
- Existential quantification: $\exists x : \Phi$

Intermezzo – First-order Logic – Syntax

Term t is defined as:

- Variable: x
- Compound: $f(t_1, \ldots, t_n)$

An FO formula is defined as:

- Atom: $P(t_1, \ldots, t_n)$
- Negation: $\neg\phi$
- Disjunction: $\phi_1 \vee \phi_2$
- Existential quantification: $\exists x : \Phi$

Intermezzo – First-order Logic – Example

- Natural language:

  Every node in a graph has at least one outgoing edge.

- In first-order logic:

$$\forall x : \mathit{Node}(x) \Rightarrow \exists y : \mathit{Node}(y) \wedge \mathit{Edge}(x, y)$$

Intermezzo – First-order Logic – Example

- Natural language:

  Every node in a graph has at least one outgoing edge.

- In first-order logic:

$$\forall x : \mathit{Node}(x) \Rightarrow \exists y : \mathit{Node}(y) \wedge \mathit{Edge}(x, y)$$

[Satisfaction or truth relation] Let ϕ be a formula interpreted by $\mathfrak{A}$. We define that $\mathfrak{A}$ satisfies ϕ (denoted $\mathfrak{A} \models \phi$) by an inductive case analysis on the structure of ϕ:

- $\mathfrak{A} \models P(t_1, \ldots, t_n)$ if $(t_1^{\mathfrak{A}}, \ldots, t_n^{\mathfrak{A}}) \in P$;
- $\mathfrak{A} \models (\neg\alpha)$ if $\mathfrak{A} \not\models \alpha$; (that is, $\mathfrak{A}$ does not satisfy α);
- $\mathfrak{A} \models (\alpha \vee \beta)$ if $\mathfrak{A} \models \alpha$ or $\mathfrak{A} \models \beta$ (or both);
- $\mathfrak{A} \models (\exists x : \alpha)$ if there exists $d \in D^{\mathfrak{A}}$ such that $\mathfrak{A}[x : d] \models \alpha$;

Here D is denoting the domain od discourse, $s^{\mathfrak{A}}$ the value of symbol s in the structure $\mathfrak{A}$, and $\mathfrak{A}[x : d]$ extended structure with value d for symbol x.

[Satisfaction or truth relation] Let ϕ be a formula interpreted by $\mathfrak{A}$. We define that $\mathfrak{A}$ satisfies ϕ (denoted $\mathfrak{A} \models \phi$) by an inductive case analysis on the structure of ϕ:

- $\mathfrak{A} \models P(t_1, \ldots, t_n)$ if $(t_1^{\mathfrak{A}}, \ldots, t_n^{\mathfrak{A}}) \in P$;
- $\mathfrak{A} \models (\neg\alpha)$ if $\mathfrak{A} \not\models \alpha$; (that is, $\mathfrak{A}$ does not satisfy α);
- $\mathfrak{A} \models (\alpha \vee \beta)$ if $\mathfrak{A} \models \alpha$ or $\mathfrak{A} \models \beta$ (or both);
- $\mathfrak{A} \models (\exists x : \alpha)$ if there exists $d \in D^{\mathfrak{A}}$ such that $\mathfrak{A}[x : d] \models \alpha$;

Here D is denoting the domain od discourse, $s^{\mathfrak{A}}$ the value of symbol s in the structure $\mathfrak{A}$, and $\mathfrak{A}[x : d]$ extended structure with value d for symbol x.

Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

Intermezzo – First-order Logic and inductive definitions

Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

Intermezzo – First-order Logic and inductive definitions

Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

Well-founded semantics.

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

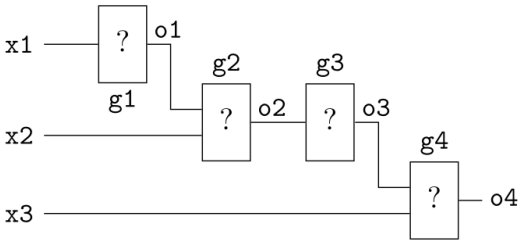
Extending logic with inductive definitions:

$$\left\{ \begin{array}{l} \forall x, y : T(x, y) \leftarrow E(x, y). \\ \forall x, y, z : T(x, z) \leftarrow T(x, y) \wedge T(y, z). \end{array} \right\}$$

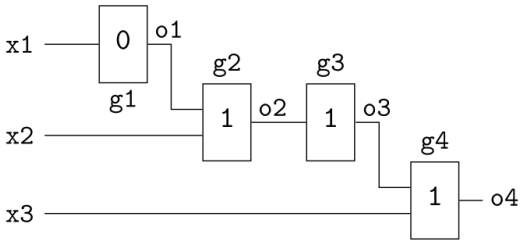
Well-founded semantics.
Easy to embed in the model semantics.

A logic of nonmonotone inductive definitions. Marc Denecker, Eugenia Ternovska. ACM Transactions on Computational Logic, 2008.

Intermezzo – Example from ICLP'24 Competition



(a) A Boolean circuit with four gates



(b) The unique solution

Solving the challenge with the IDP3 system [Demo].

ICLP'24 Programming Contest materials.

Intermezzo – Example from ICLP'24 Competition

https://idp.cs.kuleuven.be/idp/

Predicate Logic as a Modelling Language: The IDP System. Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, Gerda Janssens, Marc Denecker. Declarative Logic Programming, 2018.

? Ambiguities in Knowledge Representation

Partial functions

Decision modeling

Order-sorted Intensional logic

Partial functions

➗ Partial functions

Some examples:

- Subtraction on Natural numbers (i.e., $5 - 10$)
- Division in Natural and Real numbers (i.e., $5/0$)
- The present king of France is bald (France has no king)
- Next element of a list.

➗ Partial functions

Some examples:

- Subtraction on Natural numbers (i.e., $5 - 10$)
- Division in Natural and Real numbers (i.e., $5/0$)
- The present king of France is bald (France has no king)
- Next element of a list.

➗ Partial functions

Some examples:

- Subtraction on Natural numbers (i.e., $5 - 10$)
- Division in Natural and Real numbers (i.e., $5/0$)
- The present king of France is bald (France has no king)
- Next element of a list.

➗ Partial functions

Some examples:

- Subtraction on Natural numbers (i.e., $5 - 10$)
- Division in Natural and Real numbers (i.e., $5/0$)
- The present king of France is bald (France has no king)
- Next element of a list.

➗ Partial functions

Some examples:

- Subtraction on Natural numbers (i.e., $5 - 10$)
- Division in Natural and Real numbers (i.e., $5/0$)
- The present king of France is bald (France has no king)
- Next element of a list.

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- Who thinks the statement it is **true**? ✋
- Who thinks the statement it is **false**?
- Who thinks the statement it is **neither** true or false?

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- Who thinks the statement it is **true**?
- Who thinks the statement it is **false**? ✋
- Who thinks the statement it is **neither** true or false?

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- Who thinks the statement it is **true**?
- Who thinks the statement it is **false**?
- Who thinks the statement it is **neither** true or false? ✋

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- If the statement is **false**, what about:

  The present king of France is **not** bald.

  If it is false, the *law of excluded middle* fails!

- If the statement is **undefined**, (philosophically) it has no meaning!
  But the sentence obviously has a meaning!

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- If the statement is **false**, what about:

  The present king of France is **not** bald.

  If it is false, the *law of excluded middle* fails!

- If the statement is **undefined**, (philosophically) it has no meaning!
  But the sentence obviously has a meaning!

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- If the statement is **false**, what about:

  The present king of France is **not** bald.

  If it is false, the *law of excluded middle* fails!

- If the statement is **undefined**, (philosophically) it has no meaning!
  But the sentence obviously has a meaning!

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

- What is the issue with:

  The present king of France is bald.

- If the statement is **false**, what about:

  The present king of France is **not** bald.

  If it is false, the *law of excluded middle* fails!

- If the statement is **undefined**, (philosophically) it has no meaning!
  But the sentence obviously has a meaning!

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions – Analysis

How about this statement:

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

Is it true of false (or neither)?

÷ Partial functions – Analysis

Let's give it another try:

The present king of France is bald.

So far we concluded: The sentence is not false nor true, but it has a meaning.

- According to Russell the sentence means:

  There exist exactly one person that is the present king of France and that person is bald.

- Alternative?

  If a person is the present king of France then that person is bald.

Conclusion?: The sentence is ambiguous, and interpretation depends on the context.

÷ Partial functions – Analysis

Let's give it another try:

The present king of France is bald.

So far we concluded: The sentence is not false nor true, but it has a meaning.

- According to Russell the sentence means:

  There exist exactly one person that is the present king of France and that person is bald.

- Alternative?

  If a person is the present king of France then that person is bald.

Conclusion?: The sentence is ambiguous, and interpretation depends on the context.

÷ Partial functions – Analysis

Let's give it another try:

The present king of France is bald.

So far we concluded: The sentence is not false nor true, but it has a meaning.

- According to Russell the sentence means:

  There exist exactly one person that is the present king of France and that person is bald.

- Alternative?

  If a person is the present king of France then that person is bald.

Conclusion?: The sentence is ambiguous, and interpretation depends on the context.

÷ Partial functions – Analysis

Let's give it another try:

The present king of France is bald.

So far we concluded: The sentence is not false nor true, but it has a meaning.

- According to Russell the sentence means:

  There exist exactly one person that is the present king of France and that person is bald.

- Alternative?

  If a person is the present king of France then that person is bald.

Conclusion?: The sentence is ambiguous, and interpretation depends on the context.

÷ Partial functions and Knowledge Representation

How to treat partial functions in Knowledge Representation:

- There is no space for ambiguities in Knowledge Representation.
- A good KR language should eliminate ambiguities by design (opposite of the natural language).
- This should be decidable property of the language.
- We propose the solution: Guarded Partial Function Logic

÷ Partial functions and Knowledge Representation

How to treat partial functions in Knowledge Representation:

- There is no space for ambiguities in Knowledge Representation.
- A good KR language should eliminate ambiguities by design (opposite of the natural language).
- This should be decidable property of the language.
- We propose the solution: Guarded Partial Function Logic

÷ Partial functions and Knowledge Representation

How to treat partial functions in Knowledge Representation:

- There is no space for ambiguities in Knowledge Representation.
- A good KR language should eliminate ambiguities by design (opposite of the natural language).
- This should be decidable property of the language.
- We propose the solution: Guarded Partial Function Logic

÷ Partial functions and Knowledge Representation

How to treat partial functions in Knowledge Representation:

- There is no space for ambiguities in Knowledge Representation.
- A good KR language should eliminate ambiguities by design (opposite of the natural language).
- This should be decidable property of the language.
- We propose the solution: Guarded Partial Function Logic

÷ Partial functions and Knowledge Representation

How to treat partial functions in Knowledge Representation:

- There is no space for ambiguities in Knowledge Representation.
- A good KR language should eliminate ambiguities by design (opposite of the natural language).
- This should be decidable property of the language.
- We propose the solution: Guarded Partial Function Logic

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic

Basic guards:

- Each function symbol f is paired with domain predicate δ_f.
- $[f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is defined iff $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Conditional guards:

$$\textbf{if } \delta_f(t_1, \ldots, t_n) \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi}$$

- In sub-formula ϕ term $f(t_1, \ldots, t_n)$ is guarded.
- However, terms $t_1, \ldots, t_n$ are not!

Guarded Partial Function Logic – Example

Let the sentence The present king of France is bald be modeled as:

$$Bald(KoF)$$

The guarded versions of this sentence:

- King of France exist and he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } false \textbf{ fi}$$

- If the king of France exist he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } true \textbf{ fi}$$

Guarded Partial Function Logic – Example

Let the sentence The present king of France is bald be modeled as:

$$Bald(KoF)$$

The guarded versions of this sentence:

- King of France exist and he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } false \textbf{ fi}$$

- If the king of France exist he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } true \textbf{ fi}$$

Guarded Partial Function Logic – Example

Let the sentence The present king of France is bald be modeled as:

$$Bald(KoF)$$

The guarded versions of this sentence:

- King of France exist and he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } \mathit{false} \textbf{ fi}$$

- If the king of France exist he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } \mathit{true} \textbf{ fi}$$

Guarded Partial Function Logic – Example

Let the sentence The present king of France is bald be modeled as:

$$Bald(KoF)$$

The guarded versions of this sentence:

- King of France exist and he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } false \textbf{ fi}$$

- If the king of France exist he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } true \textbf{ fi}$$

Guarded Partial Function Logic – Example

Let the sentence The present king of France is bald be modeled as:

$$Bald(KoF)$$

The guarded versions of this sentence:

- King of France exist and he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } false \textbf{ fi}$$

- If the king of France exist he is bald:

$$\textbf{if } \delta_{KoF}() \textbf{ then } Bald(KoF) \textbf{ else } true \textbf{ fi}$$

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  $$\textbf{if } \delta_{mp}() \textbf{ then}$$
  $$\quad \textbf{if } \delta_{mother}(mp) \textbf{ then } Doctor(mother(mp)) \textbf{ else } \mathit{false} \textbf{ fi}$$
  $$\textbf{else } \mathit{false} \textbf{ fi}.$$

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  if $\delta_{mp}()$ **then**
  if $\delta_{mother}(mp)$ **then** $Doctor(mother(mp))$ **else** *false* **fi**
  else *false* **fi**.

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  $$\textbf{if } \delta_{mp}() \textbf{ then}$$
  $$\quad \textbf{if } \delta_{mother}(mp) \textbf{ then } Doctor(mother(mp)) \textbf{ else } false \textbf{ fi}$$
  $$\textbf{else } false \textbf{ fi}.$$

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  if $\delta_{mp}()$ **then**
  if $\delta_{mother}(mp)$ **then** $Doctor(mother(mp))$ **else** *false* **fi**
  else *false* **fi.**

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  if $\delta_{mp}()$ **then**
  if $\delta_{mother}(mp)$ **then** $Doctor(mother(mp))$ **else** *false* **fi**
  else *false* **fi.**

Guarded Partial Function Logic

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

  $$Doctor(mother(mp))$$

- Guarded:

  if $\delta_{mp}()$ **then**
  if $\delta_{mother}(mp)$ **then** $Doctor(mother(mp))$ **else** *false* **fi**
  else *false* **fi.**

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

$$\phi \mathbin{\overrightarrow{\wedge}} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \textit{false} \textbf{ fi}$$
$$\phi \mathbin{\Rrightarrow} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \textit{true} \textbf{ fi}$$

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

$$\phi \overrightarrow{\wedge} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{false} \textbf{ fi}$$
$$\phi \Rrightarrow \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{true} \textbf{ fi}$$

Example:

$$\delta_{mp}() \overrightarrow{\wedge} \delta_{mother}(mp) \overrightarrow{\wedge} Doctor(mother(mp))$$

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

$$\phi \overrightarrow{\wedge} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{false} \textbf{ fi}$$
$$\phi \Rrightarrow \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{true} \textbf{ fi}$$

Implicit guards (annotations):

$$[\![\mathit{Doctor}(\mathit{mother}(\mathit{mp}))]\!] \doteq \delta_{\mathit{mp}}() \overrightarrow{\wedge} \delta_{\mathit{mother}}(\mathit{mp}) \overrightarrow{\wedge} \mathit{Doctor}(\mathit{mother}(\mathit{mp}))$$

$$\langle\!\langle \mathit{QoF} = \mathit{wife}(\mathit{KoF}) \rangle\!\rangle \doteq \delta_{\mathit{QoF}}() \overrightarrow{\wedge} \delta_{\mathit{KoF}}() \overrightarrow{\wedge} \delta_{\mathit{wife}}(\mathit{KoF}) \Rrightarrow \mathit{QoF} = \mathit{wife}(\mathit{KoF})$$

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

$$\phi \overrightarrow{\wedge} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{false} \textbf{ fi}$$
$$\phi \Rrightarrow \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \mathit{true} \textbf{ fi}$$

Implicit guards (annotations):

$$[\![\mathit{Doctor}(\mathit{mother}(\mathit{mp}))]\!] \doteq \delta_{\mathit{mp}}() \overrightarrow{\wedge} \delta_{\mathit{mother}}(\mathit{mp}) \overrightarrow{\wedge} \mathit{Doctor}(\mathit{mother}(\mathit{mp}))$$

$$\langle\langle \mathit{QoF} = \mathit{wife}(\mathit{KoF}) \rangle\rangle \doteq \delta_{\mathit{QoF}}() \overrightarrow{\wedge} \delta_{\mathit{KoF}}() \overrightarrow{\wedge} \delta_{\mathit{wife}}(\mathit{KoF}) \Rrightarrow \mathit{QoF} = \mathit{wife}(\mathit{KoF})$$

Guarded Partial Function Logic – Conveniences

Term guards:

$$\Delta(\mathit{mother}(\mathit{mp})) \doteq \delta_{\mathit{mp}}() \overrightarrow{\wedge} \delta_{\mathit{mother}}(\mathit{mp})$$

Example:

$$\Delta(\mathit{mother}(\mathit{mp})) \overrightarrow{\wedge} \mathit{Doctor}(\mathit{mother}(\mathit{mp})).$$

Formally defined as:

$$\Delta(x) \doteq \mathit{true}$$
$$\Delta(f(s_1, \ldots, s_n)) \doteq \Delta(s_1) \overrightarrow{\wedge} \ldots \overrightarrow{\wedge} \Delta(s_n) \overrightarrow{\wedge} \delta_f(s_1, \ldots, s_n)$$

Guarded Partial Function Logic – Conveniences

Term guards:

$$\Delta(mother(mp)) \doteq \delta_{mp}() \overrightarrow{\wedge} \delta_{mother}(mp)$$

Example:

$$\Delta(mother(mp)) \overrightarrow{\wedge} Doctor(mother(mp)).$$

Formally defined as:

$$\Delta(x) \doteq true$$
$$\Delta(f(s_1, \ldots, s_n)) \doteq \Delta(s_1) \overrightarrow{\wedge} \ldots \overrightarrow{\wedge} \Delta(s_n) \overrightarrow{\wedge} \delta_f(s_1, \ldots, s_n)$$

Guarded Partial Function Logic – Conveniences

Term guards:

$$\Delta(mother(mp)) \doteq \delta_{mp}() \overrightarrow{\wedge} \delta_{mother}(mp)$$

Example:

$$\Delta(mother(mp)) \overrightarrow{\wedge} Doctor(mother(mp)).$$

Formally defined as:

$$\Delta(x) \doteq true$$
$$\Delta(f(s_1, \ldots, s_n)) \doteq \Delta(s_1) \overrightarrow{\wedge} \ldots \overrightarrow{\wedge} \Delta(s_n) \overrightarrow{\wedge} \delta_f(s_1, \ldots, s_n)$$

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- We define well-guarded formulae.
- All convenient guards are defined in terms of conditional guards.
- **Theorem 1** - Every well-guarded formula is well-defined (i.e., true or false in all structures).
- **Theorem 2** - Complexity of deciding well-guardedness is linear relative to the size of formula.

Towards Systematic Treatment of Partial Functions in Knowledge Representation. Djordje Markovic, Maurice Bruynooghe, Marc Denecker. Logics in Artificial Intelligence JELIA 2023

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- We define well-guarded formulae.
- All convenient guards are defined in terms of conditional guards.
- **Theorem 1** - Every well-guarded formula is well-defined (i.e., true or false in all structures).
- **Theorem 2** - Complexity of deciding well-guardedness is linear relative to the size of formula.

Towards Systematic Treatment of Partial Functions in Knowledge Representation. Djordje Markovic, Maurice Bruynooghe, Marc Denecker. Logics in Artificial Intelligence JELIA 2023

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- We define well-guarded formulae.
- All convenient guards are defined in terms of conditional guards.
- **Theorem 1** - Every well-guarded formula is well-defined (i.e., true or false in all structures).
- **Theorem 2** - Complexity of deciding well-guardedness is linear relative to the size of formula.

Towards Systematic Treatment of Partial Functions in Knowledge Representation. Djordje Markovic, Maurice Bruynooghe, Marc Denecker. Logics in Artificial Intelligence JELIA 2023

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- We define well-guarded formulae.
- All convenient guards are defined in terms of conditional guards.
- **Theorem 1** - Every well-guarded formula is well-defined (i.e., true or false in all structures).
- **Theorem 2** - Complexity of deciding well-guardedness is linear relative to the size of formula.

Towards Systematic Treatment of Partial Functions in Knowledge Representation. Djordje Markovic, Maurice Bruynooghe, Marc Denecker. Logics in Artificial Intelligence JELIA 2023

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- We define well-guarded formulae.
- All convenient guards are defined in terms of conditional guards.
- **Theorem 1** - Every well-guarded formula is well-defined (i.e., true or false in all structures).
- **Theorem 2** - Complexity of deciding well-guardedness is linear relative to the size of formula.

Towards Systematic Treatment of Partial Functions in Knowledge Representation. Djordje Markovic, Maurice Bruynooghe, Marc Denecker. Logics in Artificial Intelligence JELIA 2023

Guarded Partial Function Logic - Under revision

Current research (under revision):

- We extend the notion of guards to the theory (i.e., set of logical sentences).

$$\delta_{mp}().$$
$$\delta_{mother}(mp).$$
$$Doctor(mother(mp)).$$

- Support for recursive definitions of functions.

$$\{\ \forall x, y : spouse(x) = y \leftarrow Married(x, y)\ \}$$

A Prudent Logic of Partial Functions. Djordje Markovic, Robbe Van den Eede, Marc Denecker. Under consideration for publication in the Annals of Mathematics and Artificial Intelligence.

Guarded Partial Function Logic - Under revision

Current research (under revision):

- We extend the notion of guards to the theory (i.e., set of logical sentences).

$$\delta_{mp}().$$
$$\delta_{mother}(mp).$$
$$Doctor(mother(mp)).$$

- Support for recursive definitions of functions.

$$\{\ \forall x, y : spouse(x) = y \leftarrow Married(x, y)\ \}$$

A Prudent Logic of Partial Functions. Djordje Markovic, Robbe Van den Eede, Marc Denecker. Under consideration for publication in the Annals of Mathematics and Artificial Intelligence.

? Ambiguities in Knowledge Representation

~~Partial functions~~

Decision modeling

Order-sorted Intensional logic

Decision modeling

❓ Quick poll

How many of you are familiar with/have heard of (raise your hand 👋):

- Rule-based decision modeling?
- DMN, OpenRules?
- Epistemic logic?

Quick poll

How many of you are familiar with/have heard of :

- Rule-based decision modeling? ✋
- DMN, OpenRules?
- Epistemic logic?

Quick poll

How many of you are familiar with/have heard of :

- Rule-based decision modeling?
- DMN, OpenRules? ✋
- Epistemic logic?

Quick poll

How many of you are familiar with/have heard of :

- Rule-based decision modeling?
- DMN, OpenRules?
- Epistemic logic? ✋

Decision modeling – Introduction

Rule-based decision modeling with Decision Model and Notation (DMN):

Categorizing clients				
U	Client type	On deposit	Estimaded Net Worth	Client category
1	Business	< 100000	High	High value Business
2	Business	≥ 100000	not(High)	High value Business
3	Business	< 100000	not(High)	Business Standard
4	Private	≥ 20000	High	Personal Wealth Management
5	Private	≥ 20000	not(High)	Personal Wealth Management
6	Private	< 20000	-	Personal standard

Wikipedia example. `https://commons.wikimedia.org/wiki/File:DMN_client_category_table.jpg`

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

  ```
  Good morning, Ms. Smith,
  We are happy to announce our special offer for the new GPU card.
  ```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
`https://dmcommunity.org/challenge/challenge-aug-2016/`

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

```
Good morning, Ms.  Smith,
We are happy to announce our special offer for the new GPU card.
```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
`https://dmcommunity.org/challenge/challenge-aug-2016/`

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

```
Good morning, Ms.  Smith,
We are happy to announce our special offer for the new GPU card.
```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
https://dmcommunity.org/challenge/challenge-aug-2016/

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

```
Good morning, Ms. Smith,
We are happy to announce our special offer for the new GPU card.
```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
https://dmcommunity.org/challenge/challenge-aug-2016/

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

```
Good morning, Ms.  Smith,
We are happy to announce our special offer for the new GPU card.
```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
https://dmcommunity.org/challenge/challenge-aug-2016/

Decision modeling – Example

- Greeting a Customer with Unknown Data[1]
- Context: Composing newsletters for a company.
- For example:

```
Good morning, Ms.  Smith,
We are happy to announce our special offer for the new GPU card.
```

- Greeting
- Salutation
- Message

[1] *Challenge appeared in 2016 as part of the Decision Management Community*
https://dmcommunity.org/challenge/challenge-aug-2016/

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown

Input parameters:

- Time at the user's location [00 – 23]
- Summer/Winter time at the user's location [Summer, Winter]
- The gender of the user [Male, Female]
- The marital status of the user [Married, Single]
- The GPU that the user owns [GA, GB, GC, GD]

Output parameters:

- Greeting [Good Morning, G. Afternoon, G. Evening, G. Night]
- Salutation [Mr, Ms, Mrs]
- Message [M1, M2, M3, M4]

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Greeting:

• "Good Morning"	Summer [00..11) or Winter [00..12)
• "Good Afternoon"	Summer [11..17) or Winter [12..16)
• "Good Evening"	Summer [17..22) or Winter [16..21)
• "Good Night"	Summer [22..24) or Winter [21..24)
• Same greeting	If the time is in the range such that Summer/Winter time is irrelevant
• "Hello"	If no sufficient information

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Greeting:

• "Good Morning"	Summer [00..11) or Winter [00..12)
• "Good Afternoon"	Summer [11..17) or Winter [12..16)
• "Good Evening"	Summer [17..22) or Winter [16..21)
• "Good Night"	Summer [22..24) or Winter [21..24)
• Same greeting	If the time is in the range such that Summer/Winter time is irrelevant
• "Hello"	If no sufficient information

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Greeting:

• "Good Morning"	Summer [00..11) or Winter [00..12)
• "Good Afternoon"	Summer [11..17) or Winter [12..16)
• "Good Evening"	Summer [17..22) or Winter [16..21)
• "Good Night"	Summer [22..24) or Winter [21..24)
• Same greeting	If the time is in the range such that Summer/Winter time is irrelevant
• "Hello"	If no sufficient information

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Salutation:

- "Mr" — Male
- "Ms" — Single, Female
- "Mrs" — Married, Female
- "Ms" — If gender is known to be Female and marital status is unknown
- "Customer" — If the gender is unknown

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Salutation:

• "Mr"	Male
• "Ms"	Single, Female
• "Mrs"	Married, Female
• "Ms"	If gender is known to be Female and marital status is unknown
• "Customer"	If the gender is unknown

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Salutation:

• "Mr"	Male
• "Ms"	Single, Female
• "Mrs"	Married, Female
• "Ms"	If gender is known to be Female and marital status is unknown
• "Customer"	If the gender is unknown

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Message:

- Msg1 (performance comparison) – Exact GPU card of the customer is known.
- Msg2 (form for performance comparison) – It is known that the customer has a GPU card, but it is not known which one.
- Msg3 (discount offer) – It is known that the customer does not have a GPU card.
- Msg4 (send a poll) – It is not known whether the customer has GPU.

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Message:

- Msg1 (performance comparison) – Exact GPU card of the customer is known.
- Msg2 (form for performance comparison) – It is known that the customer has a GPU card, but it is not known which one.
- Msg3 (discount offer) – It is known that the customer does not have a GPU card.
- Msg4 (send a poll) – It is not known whether the customer has GPU.

Decision modeling – Greeting a Customer with Unknown Data

Decision rules:

Deciding the Message:

- Msg1 (performance comparison) – Exact GPU card of the customer is known.
- Msg2 (form for performance comparison) – It is known that the customer has a GPU card, but it is not known which one.
- Msg3 (discount offer) – It is known that the customer does not have a GPU card.
- Msg4 (send a poll) – It is not known whether the customer has GPU.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Decision modeling – Greeting a Customer with Unknown Data

Problems:

- Missing/Unknown data:
  - Some decisions can be made even with incomplete information.
  - E.g., Salutation is "Mr" regardless of marital status (if gender is male).
  - E.g., it suffices to know that the time is between 00 and 11 (if Summer time) to decide that Greeting "Good morning" should be used.
- Non-existing values:
  - Some variables do not have a value.
  - A PC does not have to have a GPU card.
  - This is different from existing but unknown values.

Existing rule-based decision modeling languages do not entirely support unknown and non-existing values leading to semantic mismatch.

Decision modeling – Greeting a Customer with Unknown Data

Attempt to model salutation in pure DMN:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms

Decision modeling – Greeting a Customer with Unknown Data

Attempt to model salutation in pure DMN:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms

Is this rule expressing objective or epistemic relation?

Decision modeling – epistemic DMN

Epistemic DMN (idea):

- Conditions of the rules are interpreted epistemically.
- Constants can be partial (i.e., non-denoting).
- Observation: There is a hierarchy in decision making.

An epistemic logic for modeling decisions in the context of incomplete knowledge. Djordje Markovic, Simon Vandevelde, Linde Vanbesien, Joost Vennekens, Marc Denecker. SAC'24: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing 2024

Decision modeling – epistemic DMN

Epistemic DMN (idea):

- Conditions of the rules are interpreted epistemically.
- Constants can be partial (i.e., non-denoting).
- Observation: There is a hierarchy in decision making.

An epistemic logic for modeling decisions in the context of incomplete knowledge. Djordje Markovic, Simon Vandevelde, Linde Vanbesien, Joost Vennekens, Marc Denecker. SAC'24: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing 2024

Decision modeling – epistemic DMN

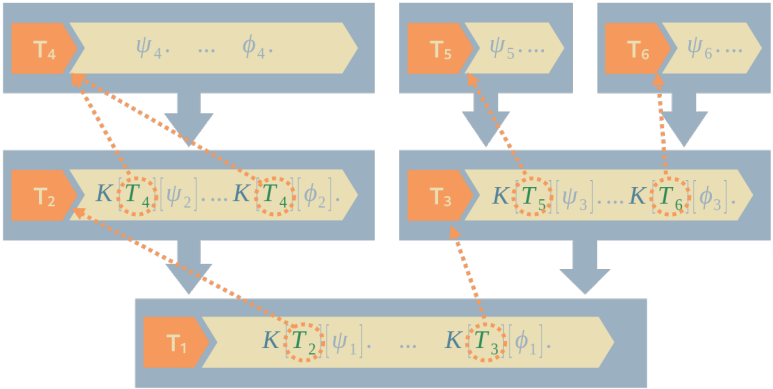
Epistemic DMN (idea):

- Conditions of the rules are interpreted epistemically.
- Constants can be partial (i.e., non-denoting).
- Observation: There is a hierarchy in decision making.

An epistemic logic for modeling decisions in the context of incomplete knowledge. Djordje Markovic, Simon Vandevelde, Linde Vanbesien, Joost Vennekens, Marc Denecker. SAC'24: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing 2024

Decision modeling – Ordered Epistemic Logic

Ordered Epistemic Logic:



Ordered Epistemic Logic: Semantics, Complexity and Applications. Hanne Vlaeminck, Joost Vennekens, Maurice Bruynooghe, Marc Denecker. KR (2012): Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning.

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Types		
Name	**DataType**	**Possible Values**
time	Int	[0..23]
sw_time	String	Summer, Winter
greeting	String	Good Morning, Good Afternoon, Good Evening, Good Night, Hello
gender	String	Female, Male
marital status	String	Single, Married
salutation	String	Mr, Mrs, Ms, Customer
gpu	String	GA, GB, GC, GD
message	String	Msg1, Msg2, Msg3, Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Types		
Name	**DataType**	**Possible Values**
time	Int	[0..23]
sw_time	String	Summer, Winter
greeting	String	Good Morning, Good Afternoon, Good Evening, Good Night, Hello
gender	String	Female, Male
marital status	String	Single, Married
salutation	String	Mr, Mrs, Ms, Customer
gpu	String	GA, GB, GC, GD
message	String	Msg1, Msg2, Msg3, Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Types		
Name	**DataType**	**Possible Values**
time	Int	[0..23]
sw_time	String	Summer, Winter
greeting	String	Good Morning, Good Afternoon, Good Evening, Good Night, Hello
gender	String	Female, Male
marital status	String	Single, Married
salutation	String	Mr, Mrs, Ms, Customer
gpu	String	GA, GB, GC, GD
message	String	Msg1, Msg2, Msg3, Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Types		
Name	**DataType**	**Possible Values**
time	Int	[0..23]
sw_time	String	Summer, Winter
greeting	String	Good Morning, Good Afternoon, Good Evening, Good Night, Hello
gender	String	Female, Male
marital status	String	Single, Married
salutation	String	Mr, Mrs, Ms, Customer
gpu	String	GA, GB, GC, GD
message	String	Msg1, Msg2, Msg3, Msg4

Modeling the ontology:

Constants	
Name	**DataType**
Time	time
SW Time	sw_time
Greeting	greeting
Gender	gender
M Status	marital status
Salutation	salutation
Message	message
Partial GPU	gpu

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Constants	
Name	**DataType**
Time	time
SW Time	sw_time
Greeting	greeting
Gender	gender
M Status	marital status
Salutation	salutation
Message	message
Partial GPU	gpu

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Constants	
Name	**DataType**
Time	time
SW Time	sw_time
Greeting	greeting
Gender	gender
M Status	marital status
Salutation	salutation
Message	message
Partial GPU	gpu

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the ontology:

Constants	
Name	**DataType**
Time	time
SW Time	sw_time
Greeting	greeting
Gender	gender
M Status	marital status
Salutation	salutation
Message	message
Partial GPU	gpu

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Greeting decision:

Greeting			
F	Time	SW Time	Greeting
1	[0..11)	*Summer*	Good Morning
2	[11..17)	*Summer*	Good Afternoon
3	[17..22)	*Summer*	Good Evening
4	[22..24)	*Summer*	Good Night
5	[0..12)	*Winter*	Good Morning
6	[12..16)	*Winter*	Good Afternoon
7	[16..21)	*Winter*	Good Evening
8	[21..24)	*Winter*	Good Night
9	[0..11)	$\neg\lvert K\rvert$	Good Morning
10	[12..16)	$\neg\lvert K\rvert$	Good Afternoon
11	[17..21)	$\neg\lvert K\rvert$	Good Evening
12	[22..24)	$\neg\lvert K\rvert$	Good Night
13	-	-	Hello

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Salutation decision:

Salutation			
U	Gender	M Status	Salutation
1	Male	-	Mr
2	Female	Married	Mrs
3	Female	Single	Ms
4	Female	$\neg\lvert K\rvert$	Ms
5	$\neg\lvert K\rvert$	-	Customer

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\lvert K \rvert$	Msg1
2	Def	$\neg\lvert K \rvert$	Msg2
3	Undef	-	Msg3
4	$\neg\lvert K \rvert$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D GPU	GPU	Message
1	Def	$\lvert K \rvert$	Msg1
2	Def	$\neg\lvert K \rvert$	Msg2
3	Undef	-	Msg3
4	$\neg\lvert K \rvert$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\lvert K \rvert$	Msg1
2	Def	$\neg\lvert K \rvert$	Msg2
3	Undef	-	Msg3
4	$\neg\lvert K \rvert$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\lvert K \rvert$	Msg1
2	Def	$\neg\lvert K \rvert$	Msg2
3	Undef	-	Msg3
4	$\neg\lvert K \rvert$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\|K\|$	Msg1
2	Def	$\neg\|K\|$	Msg2
3	Undef	-	Msg3
4	$\neg\|K\|$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\|K\|$	Msg1
2	Def	$\neg\|K\|$	Msg2
3	Undef	-	Msg3
4	$\neg\|K\|$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\lvert K \rvert$	Msg1
2	Def	$\neg\lvert K \rvert$	Msg2
3	Undef	-	Msg3
4	$\neg\lvert K \rvert$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the Message decision:

Message			
U	D_GPU	GPU	Message
1	Def	$\|K\|$	Msg1
2	Def	$\neg\|K\|$	Msg2
3	Undef	-	Msg3
4	$\neg\|K\|$	-	Msg4

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the epistemic state:

K. Time	
K	Time
1	[8..10]

K. SW	
K	SW Time
1	Summer

K. Gender	
K	Gender
1	Male

K. D_GPU	
K	D_GPU
1	Def

K. GPU	
K	GPU
1	GA,GB

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the epistemic state:

K. Time	
K	Time
1	[8..10]

K. SW	
K	SW Time
1	Summer

K. Gender	
K	Gender
1	Male

K. D_GPU	
K	D_GPU
1	Def

K. GPU	
K	GPU
1	GA,GB

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the epistemic state:

K. Time	
K	Time
1	[8..10]

K. SW	
K	SW Time
1	Summer

K. Gender	
K	Gender
1	Male

K. D_GPU	
K	D_GPU
1	Def

K. GPU	
K	GPU
1	GA,GB

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the epistemic state:

K. Time	
K	Time
1	[8..10]

K. SW	
K	SW Time
1	Summer

K. Gender	
K	Gender
1	Male

K. D_GPU	
K	D_GPU
1	Def

K. GPU	
K	GPU
1	GA,GB

Decision modeling – Solving Greeting a Customer with eDMN

Modeling the epistemic state:

K. Time	
K	Time
1	[8..10]

K. SW	
K	SW Time
1	Summer

K. Gender	
K	Gender
1	Male

K. D_GPU	
K	D_GPU
1	Def

K. GPU	
K	GPU
1	GA,GB

Decision modeling – Solving Greeting a Customer with eDMN

Implementation based on the IDP-Z3 KB system.

https://www.idp-z3.be/

https://gitlab.com/krr/idp-z3-oel

? Ambiguities in Knowledge Representation

~~Partial functions~~

~~Decision modeling~~

Order-sorted Intensional logic

Order-sorted Intensional logic

Order-sorted Intensional logic

- Consider the natural language sentence:

  An animal produces sound iff it produces a sound characteristic for its species.

- What we mean:

  An animal *produces sound* iff
  that animal is a dog and it barks,
  or that animal is a cat and it meows, ...

Order-sorted Intensional logic

- Consider the natural language sentence:

  An animal produces sound iff it produces a sound characteristic for its species.

- What we mean:

  An animal *produces sound* iff
  that animal is a dog and it barks,
  or that animal is a cat and it meows, ...

Order-sorted Intensional logic

- Consider the natural language sentence:

  An animal *produces sound* iff it produces a sound characteristic for its species.

- What we mean:

  An animal *produces sound* iff
  that animal is a dog and it barks,
  or that animal is a cat and it meows, ...

Order-sorted Intensional logic

- Consider the natural language sentence:

  An animal *produces sound* iff it produces a sound characteristic for its species.

- What we mean:

  An animal *produces sound* iff
  that animal is a dog and it barks,
  or that animal is a cat and it meows, ...

Order-sorted Intensional logic

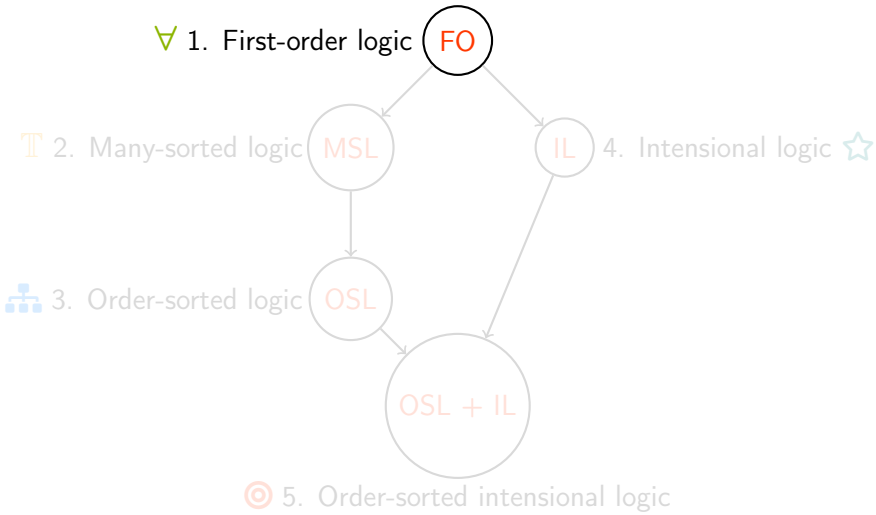
- Consider the natural language sentence:

  An animal *produces sound* iff it produces a sound characteristic for its species.

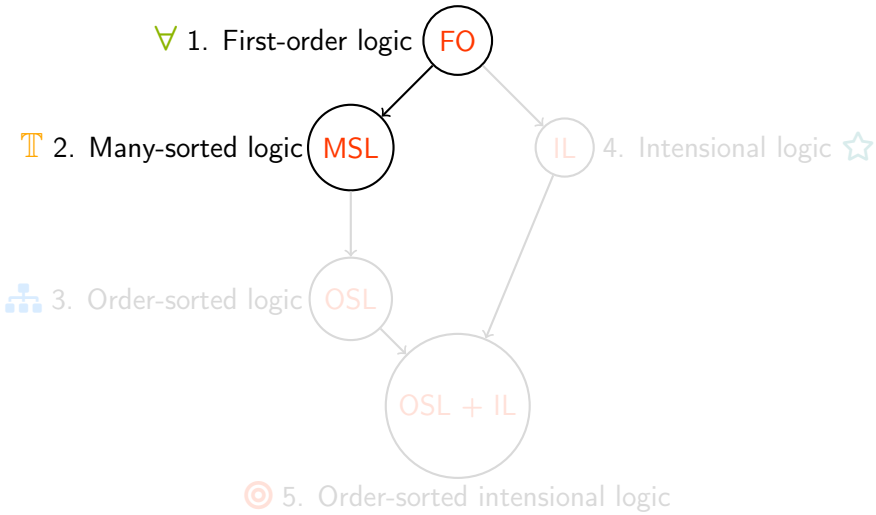
- What we mean:

  An animal *produces sound* iff
  that animal is a dog and it barks,
  or that animal is a cat and it meows, ...

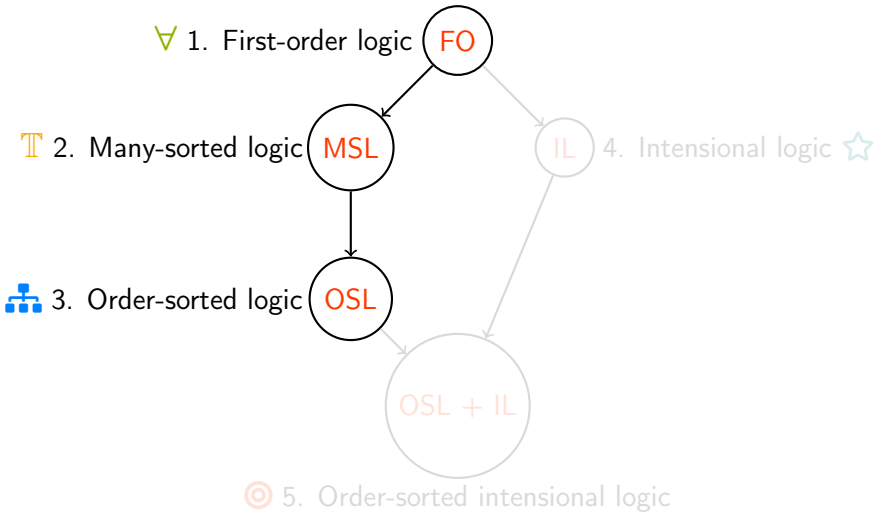
Order-sorted Intensional logic - Roadmap



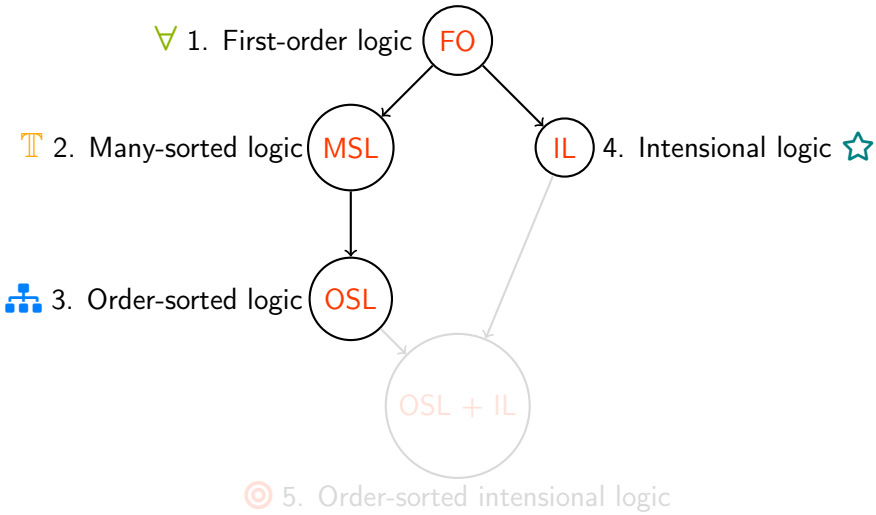
Order-sorted Intensional logic - Roadmap



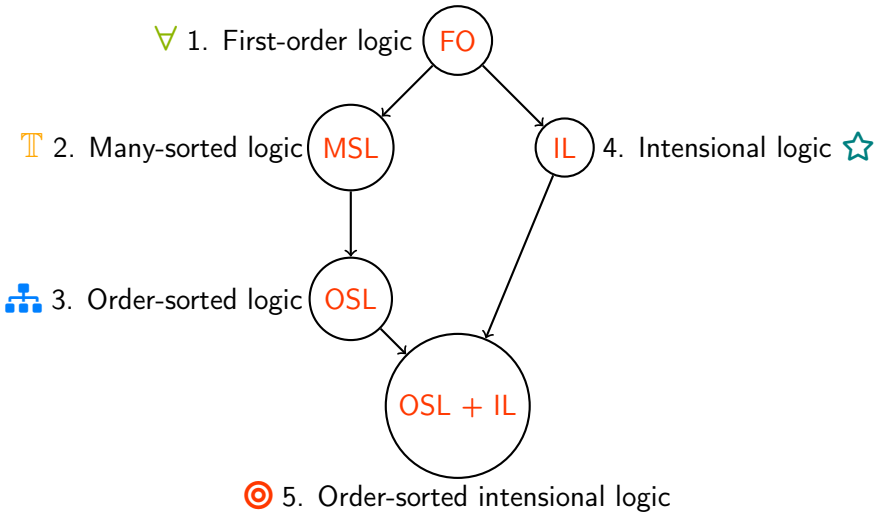
Order-sorted Intensional logic - Roadmap



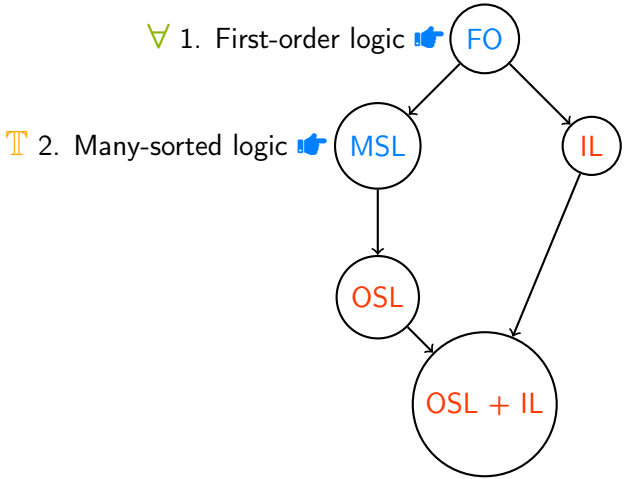
Order-sorted Intensional logic - Roadmap



Order-sorted Intensional logic - Roadmap



Order-sorted Intensional logic - Roadmap



Order-sorted Intensional logic – First-order and many-sorted logic

- Example:

"There is a *Cat* eating."

- **First-order logic** is not typed! Each object is part of the "Universe".

$$\exists c : Cat(c) \wedge Eats(c).$$

- In **many-sorted logic**, we can introduce sort/type *Cat*.
- Then we can declare eats predicate as: $Eats : (\underline{\text{Cat}}) \rightarrow \mathbb{B}$
- Finally, we express the example statement as:

$$\exists c[Cat] : Eats(\underline{c}).$$

Order-sorted Intensional logic – First-order and many-sorted logic

- Example:

  "There is a *Cat* eating."

- **First-order logic** is not typed! Each object is part of the "Universe".

$$\exists c : \mathit{Cat}(c) \wedge \mathit{Eats}(c).$$

- In **many-sorted logic**, we can introduce sort/type *Cat*.
- Then we can declare eats predicate as: $\mathit{Eats} : (\underline{\mathrm{Cat}}) \to \mathbb{B}$
- Finally, we express the example statement as:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{c}).$$

Order-sorted Intensional logic – First-order and many-sorted logic

- Example:

  "There is a *Cat* eating."

- **First-order logic** is not typed! Each object is part of the "Universe".

$$\exists c : \mathit{Cat}(c) \wedge \mathit{Eats}(c).$$

- In **many-sorted logic**, we can introduce sort/type *Cat*.
- Then we can declare eats predicate as: $\mathit{Eats} : (\underline{\mathrm{Cat}}) \to \mathbb{B}$
- Finally, we express the example statement as:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{c}).$$

Order-sorted Intensional logic – First-order and many-sorted logic

- Example:

  "There is a *Cat* eating."

- **First-order logic** is not typed! Each object is part of the "Universe".

$$\exists c : \mathit{Cat}(c) \wedge \mathit{Eats}(c).$$

- In **many-sorted logic**, we can introduce sort/type *Cat*.
- Then we can declare eats predicate as: $\mathit{Eats} : (\underline{\mathrm{Cat}}) \to \mathbb{B}$
- Finally, we express the example statement as:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{c}).$$

- Example:

"There is a *Cat* eating."

- **First-order logic** is not typed! Each object is part of the "Universe".

$$\exists c : \mathit{Cat}(c) \wedge \mathit{Eats}(c).$$

- In **many-sorted logic**, we can introduce sort/type *Cat*.
- Then we can declare eats predicate as: $\mathit{Eats} : (\underline{\mathrm{Cat}}) \to \mathbb{B}$
- Finally, we express the example statement as:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{c}).$$

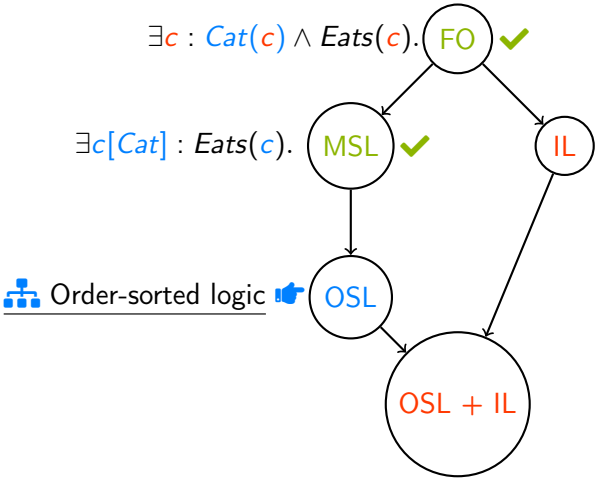
- Any problems with $\exists c[Cat] : Eats(\underline{c})$?
- How to express that there is a *Dog* eating?

$$\exists d[Dog] : Eats(\underline{d})$$

- Any problems with $\exists c[Cat] : Eats(\underline{c})$?
- How to express that there is a *Dog* eating?

$$\exists d[Dog] : Eats(\underline{d})$$

Order-sorted Intensional logic – Roadmap



Order-sorted Intensional logic – Order-sorted logic

- Hierarchy of types ($<:$ subtype relation):

$$\mathit{Dog} <: \mathit{Animal} \qquad \mathit{Cat} <: \mathit{Animal}$$

- Predicative form:

$$\mathit{Dog} : (\underline{\mathrm{Animal}}) \to \mathbb{B}$$

Order-sorted Intensional logic – Order-sorted logic

- Hierarchy of types ($<:$ subtype relation):

$$\mathit{Dog} <: \mathit{Animal} \qquad \mathit{Cat} <: \mathit{Animal}$$

- Predicative form:

$$\mathit{Dog} : (\underline{\text{Animal}}) \to \mathbb{B}$$

Order-sorted Intensional logic

- Back to the eating example:

$$Dog <: Animal \qquad Cat <: Animal$$

$$Eats : (\underline{Animal}) \to \mathbb{B}$$

- "There is an *Animal* eating" is expressed as:

$$\exists a[Animal] : Eats(\underline{a}).$$

- Both of the following statements are well-typed:

$$\exists c[Cat] : Eats(\underline{c}). \qquad \exists d[Dog] : Eats(\underline{d}).$$

Order-sorted Intensional logic

- Back to the eating example:

$$\mathit{Dog} <: \mathit{Animal} \qquad \mathit{Cat} <: \mathit{Animal}$$

$$\mathit{Eats} : (\underline{\mathrm{Animal}}) \to \mathbb{B}$$

- "There is an *Animal* eating" is expressed as:

$$\exists a[\mathit{Animal}] : \mathit{Eats}(\underline{\mathrm{a}}).$$

- Both of the following statements are well-typed:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{\mathrm{c}}). \qquad \exists d[\mathit{Dog}] : \mathit{Eats}(\underline{\mathrm{d}}).$$

Order-sorted Intensional logic

- Back to the eating example:

$$\mathit{Dog} <: \mathit{Animal} \qquad \mathit{Cat} <: \mathit{Animal}$$

$$\mathit{Eats} : (\underline{\mathrm{Animal}}) \to \mathbb{B}$$

- "There is an *Animal* eating" is expressed as:

$$\exists a[\mathit{Animal}] : \mathit{Eats}(\underline{\mathrm{a}}).$$

- Both of the following statements are well-typed:

$$\exists c[\mathit{Cat}] : \mathit{Eats}(\underline{\mathrm{c}}). \qquad \exists d[\mathit{Dog}] : \mathit{Eats}(\underline{\mathrm{d}}).$$

Order-sorted Intensional logic – Order-sorted logic

- Any problems?
- Given predicates:

$$\mathit{Bark} : (\underline{\mathrm{Dog}}) \to \mathbb{B} \qquad \mathit{Meow} : (\underline{\mathrm{Cat}}) \to \mathbb{B}$$

- Can we define the predicate:

$$\mathit{ProducingSound}(\underline{\mathrm{Animal}}) \to \mathbb{B}$$

Order-sorted Intensional logic – Order-sorted logic

- Any problems?
- Given predicates:

$$\mathit{Bark} : (\underline{\text{Dog}}) \to \mathbb{B} \qquad \mathit{Meow} : (\underline{\text{Cat}}) \to \mathbb{B}$$

- Can we define the predicate:

$$\mathit{ProducingSound}(\underline{\text{Animal}}) \to \mathbb{B}$$

Order-sorted Intensional logic – Order-sorted logic

- Any problems?
- Given predicates:

$$\mathit{Bark} : (\underline{\text{Dog}}) \to \mathbb{B} \qquad \mathit{Meow} : (\underline{\text{Cat}}) \to \mathbb{B}$$

- Can we define the predicate:

$$\mathit{ProducingSound}(\underline{\text{Animal}}) \to \mathbb{B}$$

Order-sorted Intensional logic – Order-sorted logic

- Yes we can!

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow$$
$$(\exists d[Dog] : \underline{a} =^{*1} \underline{d} \wedge Bark(\underline{d})) \vee (\exists c[Cat] : \underline{a} =^{*2} \underline{c} \wedge Meow(\underline{c})).$$

- Can we do better?

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow$$
$$(Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

Order-sorted Intensional logic – Order-sorted logic

- Yes we can!

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow$$
$$(\exists d[Dog] : \underline{a} =^{*1} \underline{d} \wedge Bark(\underline{d})) \vee (\exists c[Cat] : \underline{a} =^{*2} \underline{c} \wedge Meow(\underline{c})).$$

- Can we do better?

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow$$
$$(Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

Order-sorted Intensional logic – Order-sorted logic

- Let: $S <: T$ $S : (\underline{\mathtt{I}}) \to \mathbb{B}$ $a : T$
- Then: $S(a) \Rightarrow \phi[a]$ and $S(a) \wedge \phi[a]$ are well typed formulas!
- Abbreviated: $\langle\langle \phi(a) \rangle\rangle$ and $[[\phi(a)]]$
- Example:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow (Dog(\underline{\mathtt{a}}) \wedge Bark(\underline{\mathtt{a}})) \vee (Cat(\underline{\mathtt{a}}) \wedge Meow(\underline{\mathtt{a}})).$$

- The compact version:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow: [[Bark(\underline{\mathtt{a}})]] \vee [[Meow(\underline{\mathtt{a}})]].$$

Order-sorted Intensional logic – Order-sorted logic

- Let: $S <: T$ $S : (\underline{\mathtt{I}}) \to \mathbb{B}$ $a : T$
- Then: $S(a) \Rightarrow \phi[a]$ and $S(a) \wedge \phi[a]$ are well typed formulas!
- Abbreviated: $\langle\langle \phi(a) \rangle\rangle$ and $[[\phi(a)]]$
- Example:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow (Dog(\underline{\mathtt{a}}) \wedge Bark(\underline{\mathtt{a}})) \vee (Cat(\underline{\mathtt{a}}) \wedge Meow(\underline{\mathtt{a}})).$$

- The compact version:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow: [[Bark(\underline{\mathtt{a}})]] \vee [[Meow(\underline{\mathtt{a}})]].$$

Order-sorted Intensional logic – Order-sorted logic

- Let: $S <: T$ $S : (\underline{\mathtt{I}}) \to \mathbb{B}$ $a : T$
- Then: $S(a) \Rightarrow \phi[a]$ and $S(a) \wedge \phi[a]$ are well typed formulas!
- Abbreviated: $\langle\langle \phi(a) \rangle\rangle$ and $[[\phi(a)]]$
- Example:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- The compact version:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow: [[Bark(\underline{a})]] \vee [[Meow(\underline{a})]].$$

Order-sorted Intensional logic – Order-sorted logic

- Let: $S <: T$ $S : (\underline{\mathtt{I}}) \to \mathbb{B}$ $a : T$
- Then: $S(a) \Rightarrow \phi[a]$ and $S(a) \wedge \phi[a]$ are well typed formulas!
- Abbreviated: $\langle\langle \phi(a) \rangle\rangle$ and $[\![\phi(a)]\!]$

- Example:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow (Dog(\underline{\mathtt{a}}) \wedge Bark(\underline{\mathtt{a}})) \vee (Cat(\underline{\mathtt{a}}) \wedge Meow(\underline{\mathtt{a}})).$$

- The compact version:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow: [\![Bark(\underline{\mathtt{a}})]\!] \vee [\![Meow(\underline{\mathtt{a}})]\!].$$

Order-sorted Intensional logic – Order-sorted logic

- Let: $S <: T$ $S : (\underline{\mathtt{I}}) \to \mathbb{B}$ $a : T$
- Then: $S(a) \Rightarrow \phi[a]$ and $S(a) \wedge \phi[a]$ are well typed formulas!
- Abbreviated: $\langle\langle \phi(a) \rangle\rangle$ and $[\![\phi(a)]\!]$

- Example:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow (Dog(\underline{\mathtt{a}}) \wedge Bark(\underline{\mathtt{a}})) \vee (Cat(\underline{\mathtt{a}}) \wedge Meow(\underline{\mathtt{a}})).$$

- The compact version:

$$\forall a[Animal] : ProducingSound(\underline{\mathtt{a}}) \Leftrightarrow: [\![Bark(\underline{\mathtt{a}})]\!] \vee [\![Meow(\underline{\mathtt{a}})]\!].$$

Order-sorted Intensional logic – Order-sorted logic

- $\forall a[Animal] : ProducingSound(a) \Leftrightarrow: [[Bark(a)]] \vee [[Meow(a)]]$.

  An animal *produces sound* iff

  that animal is a dog and it barks, or that animal is a cat and it meows, ...

- Recall that we want:

  An animal *produces sound* iff it produces a sound characteristic for its species.

- Here: sound characteristic for species is collection of **concepts** as Barks, Meows, ...

Order-sorted Intensional logic – Order-sorted logic

- $\forall a[Animal] : ProducingSound(a) \Leftrightarrow: [[Bark(a)]] \vee [[Meow(a)]]$.

  An animal *produces sound* iff

  that animal is a dog and it barks, or that animal is a cat and it meows, ...

- Recall that we want:

  An animal *produces sound* iff it produces a sound characteristic for its species.

- Here: sound characteristic for species is collection of **concepts** as Barks, Meows, ...

Order-sorted Intensional logic – Order-sorted logic

- $\forall a[Animal] : ProducingSound(a) \Leftrightarrow: \mathbf{[[}Bark(a)\mathbf{]]} \vee \mathbf{[[}Meow(a)\mathbf{]]}$.

  An animal *produces sound* iff

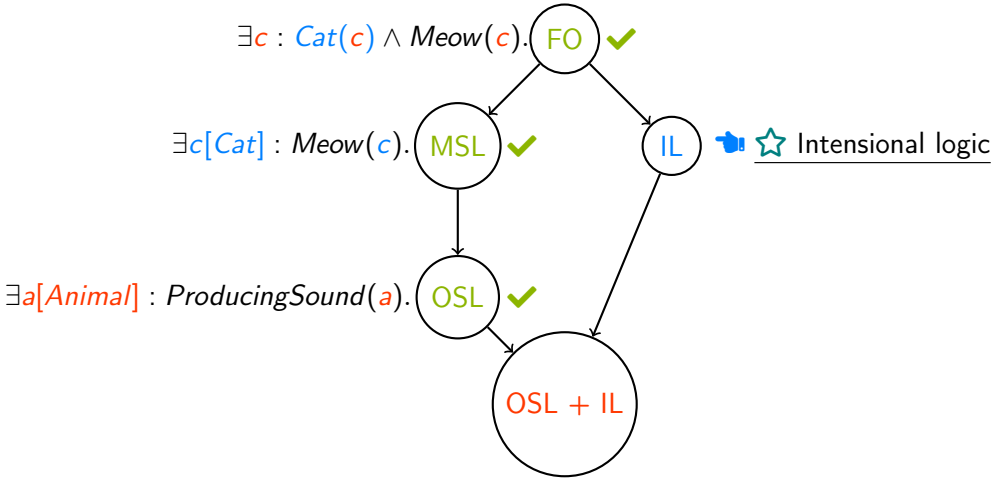
  that animal is a dog and it barks, or that animal is a cat and it meows, ...

- Recall that we want:

  An animal *produces sound* iff it produces a sound characteristic for its species.

- Here: sound characteristic for species is collection of **concepts** as Barks, Meows, ...

Order-sorted Intensional logic – Roadmap



Order-sorted Intensional logic – Intensional logic

- Extension vs Intension

- Example: Evening star and morning star are concepts.
- New type: $\mathit{Concept}$
- Elements of $\mathit{Concept} := \{\widetilde{\mathit{Animal}}, \widetilde{\mathit{Dog}}, \widetilde{\mathit{Cat}}, \widetilde{\mathit{Eats}}, \widetilde{\mathit{Bark}}, \widetilde{\mathit{Meow}}, \dots\}$
- New operators: $\text{`}(\mathit{Animal}) = \widetilde{\mathit{Animal}}$ and $\$(\widetilde{\mathit{Eats}})(d) = \mathit{Eats}(d)$
- Example:

$$\mathit{Sound}(\mathit{Concept}) \to \mathbb{B}$$

$$\mathit{Sound}(\text{`}(\mathit{Bark})) \wedge \mathit{Sound}(\text{`}(\mathit{Meow})).$$

Order-sorted Intensional logic – Intensional logic

- Extension vs Intension
- Example: Evening star and morning star are concepts.
- New type: $\mathit{Concept}$
- Elements of $\mathit{Concept} := \{\widetilde{\mathit{Animal}}, \widetilde{\mathit{Dog}}, \widetilde{\mathit{Cat}}, \widetilde{\mathit{Eats}}, \widetilde{\mathit{Bark}}, \widetilde{\mathit{Meow}}, \dots\}$
- New operators: $^\frown(\mathit{Animal}) = \widetilde{\mathit{Animal}}$ and $\$(\widetilde{\mathit{Eats}})(d) = \mathit{Eats}(d)$
- Example:

$$\mathit{Sound}(\mathit{Concept}) \to \mathbb{B}$$

$$\mathit{Sound}(^\frown(\mathit{Bark})) \wedge \mathit{Sound}(^\frown(\mathit{Meow})).$$

Order-sorted Intensional logic – Intensional logic

- Extension vs Intension
- Example: Evening star and morning star are concepts.

- New type: *Concept*

- Elements of $Concept := \{\widetilde{Animal}, \widetilde{Dog}, \widetilde{Cat}, \widetilde{Eats}, \widetilde{Bark}, \widetilde{Meow}, \dots\}$
- New operators: $^\wedge(Animal) = \widetilde{Animal}$ and $\$(\widetilde{Eats})(d) = Eats(d)$
- Example:

$$Sound(Concept) \to \mathbb{B}$$

$$Sound(^\wedge(Bark)) \wedge Sound(^\wedge(Meow)).$$

Order-sorted Intensional logic – Intensional logic

- Extension vs Intension
- Example: Evening star and morning star are concepts.
- New type: *Concept*

- Elements of *Concept* $:= \{\widetilde{Animal}, \widetilde{Dog}, \widetilde{Cat}, \widetilde{Eats}, \widetilde{Bark}, \widetilde{Meow}, \dots\}$

- New operators: $^\frown(Animal) = \widetilde{Animal}$ and $\$(\widetilde{Eats})(d) = Eats(d)$
- Example:

$$Sound(Concept) \to \mathbb{B}$$

$$Sound(^\frown(Bark)) \wedge Sound(^\frown(Meow)).$$

Order-sorted Intensional logic – Intensional logic

- Extension vs Intension
- Example: Evening star and morning star are concepts.
- New type: *Concept*
- Elements of *Concept* $:= \{\widetilde{Animal}, \widetilde{Dog}, \widetilde{Cat}, \widetilde{Eats}, \widetilde{Bark}, \widetilde{Meow}, \dots\}$
- New operators: $`(Animal) = \widetilde{Animal}$ and $\$(\widetilde{Eats})(d) = Eats(d)$
- Example:

$$Sound(Concept) \to \mathbb{B}$$

$$Sound(`(Bark)) \wedge Sound(`(Meow)).$$

Order-sorted Intensional logic – Intensional logic

- Extension vs Intension
- Example: Evening star and morning star are concepts.
- New type: *Concept*
- Elements of *Concept* $:= \{\widetilde{Animal}, \widetilde{Dog}, \widetilde{Cat}, \widetilde{Eats}, \widetilde{Bark}, \widetilde{Meow}, \dots\}$
- New operators: $`(Animal) = \widetilde{Animal}$ and $\$(\widetilde{Eats})(d) = Eats(d)$

- Example:

$$Sound(Concept) \to \mathbb{B}$$

$$Sound(`(Bark)) \wedge Sound(`(Meow)).$$

Order-sorted Intensional logic – Intensional logic

- Back to the *ProducingSound*:

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow (Dog(a) \wedge Bark(a)) \vee (Cat(a) \wedge Meow(a)).$$

- Can we do better with intentional logic?

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Concept] : Sound(s) \wedge \$(s)(a).$$

- This is not good!

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow Bark(a) \vee Meow(a).$$

- Can this be made correct in intensional logic? Yes!

Order-sorted Intensional logic – Intensional logic

- Back to the *ProducingSound*:

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow (Dog(a) \wedge Bark(a)) \vee (Cat(a) \wedge Meow(a)).$$

- Can we do better with intentional logic?

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Concept] : Sound(s) \wedge \$(s)(a).$$

- This is not good!

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow Bark(a) \vee Meow(a).$$

- Can this be made correct in intensional logic? Yes!

Order-sorted Intensional logic – Intensional logic

- Back to the *ProducingSound*:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- Can we do better with intentional logic?

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Concept] : Sound(s) \wedge \$(s)(a).$$

- This is not good!

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow Bark(\underline{a}) \vee Meow(\underline{a}).$$

- Can this be made correct in intensional logic? Yes!

Order-sorted Intensional logic – Intensional logic

- Back to the *ProducingSound*:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- Can we do better with intentional logic?

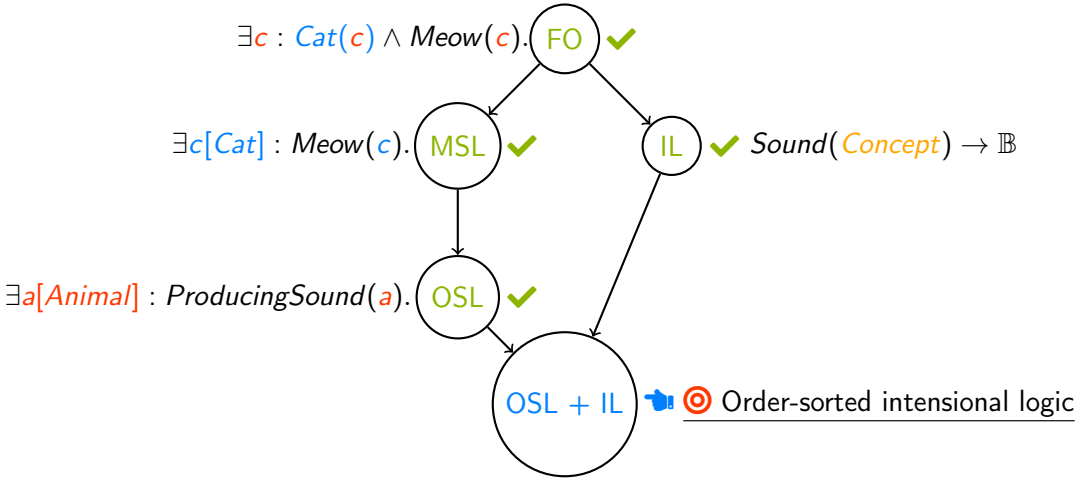
$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Concept] : Sound(s) \wedge \$(s)(a).$$

- This is not good!

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow Bark(\underline{a}) \vee Meow(\underline{a}).$$

- Can this be made correct in intensional logic? Yes!

Order-sorted Intensional logic – Roadmap



Order-sorted Intensional logic

- We defined *ProducingSound* as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- We abbreviate it with:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow [[Bark(\underline{a})]] \vee [[Meow(\underline{a})]]$$

- Introduce new type:

$$Sound <: Concept \quad := \{\widetilde{Bark}, \widetilde{Meow}\}$$

- Then *ProducingSound* can be defined as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \exists s[Sound] : [[\$(\underline{s})(a)]].$$

Order-sorted Intensional logic

- We defined *ProducingSound* as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- We abbreviate it with:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \mathbf{[[}Bark(\underline{a})\mathbf{]]} \vee \mathbf{[[}Meow(\underline{a})\mathbf{]]}$$

- Introduce new type:

$$Sound <: Concept \quad := \{\widetilde{Bark}, \widetilde{Meow}\}$$

- Then *ProducingSound* can be defined as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \exists s[Sound] : \mathbf{[[}\$(\underline{s})(a)\mathbf{]]}.$$

Order-sorted Intensional logic

- We defined *ProducingSound* as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- We abbreviate it with:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \mathbf{[[}Bark(\underline{a})\mathbf{]]} \vee \mathbf{[[}Meow(\underline{a})\mathbf{]]}$$

- Introduce new type:

$$Sound <: Concept \quad := \{\widetilde{Bark}, \widetilde{Meow}\}$$

- Then *ProducingSound* can be defined as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \exists s[Sound] : \mathbf{[[}\$(\underline{s})(a)\mathbf{]]}.$$

Order-sorted Intensional logic

- We defined *ProducingSound* as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow (Dog(\underline{a}) \wedge Bark(\underline{a})) \vee (Cat(\underline{a}) \wedge Meow(\underline{a})).$$

- We abbreviate it with:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow [[Bark(\underline{a})]] \vee [[Meow(\underline{a})]]$$

- Introduce new type:

$$Sound <: Concept \quad := \{\widetilde{Bark}, \widetilde{Meow}\}$$

- Then *ProducingSound* can be defined as:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \exists s[Sound] : [[\$(\underline{s})(a)]].$$

Order-sorted Intensional logic

- Natural language:

  An animal produces sound iff it produces a sound characteristic for its species.

- Order-sorted intensional logic:

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Sound] : [[\$(s)(a)]].$$

Order-sorted Intensional logic

- Natural language:

  An animal produces sound iff it produces a sound characteristic for its species.

- Order-sorted intensional logic:

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Sound] : [[\$(s)(a)]].$$

Order-sorted Intensional logic

- Natural language:

  An animal produces sound iff it produces a sound characteristic for its species.

- Order-sorted intensional logic:

$$\forall a[Animal] : ProducingSound(\underline{a}) \Leftrightarrow \exists s[Sound] : [[\$(\underline{s})(a)]].$$

Order-sorted Intensional logic

- Natural language:

  An animal produces sound iff it produces a sound characteristic for its species.

- Order-sorted intensional logic:

$$\forall a[Animal] : ProducingSound(a) \Leftrightarrow \exists s[Sound] : \mathbf{[[}\$(s)(a)\mathbf{]]}.$$

Order-sorted Intensional logic

- Another example:

  Fore each species there is an animal producing sound characteristic for that species.

- In order-sorted intensional logic:

$$\forall s[Sound] : \exists a[Animal] : [[\$(\underline{s})(a)]].$$

- Or grounded:

$$(\exists a[Animal] : Dog(\underline{a}) \wedge Barks(\underline{a})) \wedge (\exists a[Animal] : Cat(\underline{a}) \wedge Meow(\underline{a})).$$

Order-sorted Intensional logic

- Another example:

  Fore each species there is an animal producing sound characteristic for that species.

- In order-sorted intensional logic:

$$\forall s[Sound] : \exists a[Animal] : \mathbf{[[\$(\underline{s})(a)]]}.$$

- Or grounded:

$$(\exists a[Animal] : Dog(\underline{a}) \wedge Barks(\underline{a})) \wedge (\exists a[Animal] : Cat(\underline{a}) \wedge Meow(\underline{a})).$$

Order-sorted Intensional logic

- Another example:

  Fore each species there is an animal producing sound characteristic for that species.

- In order-sorted intensional logic:

$$\forall s[Sound] : \exists a[Animal] : \mathbf{[[\$(\underline{s})(a)]]}.$$

- Or grounded:

$$(\exists a[Animal] : Dog(\underline{a}) \wedge Barks(\underline{a})) \wedge (\exists a[Animal] : Cat(\underline{a}) \wedge Meow(\underline{a})).$$

Order-sorted Intensional logic – Conclusion

- Expressing subtyping polymorphism is a challenge in order-sorted logic.
- Intensional logic is required.
- Implicit guarding makes it compact and elegant.
- Complex well-typing relation (future work).

Order-sorted Intensional logic – Conclusion

- Expressing subtyping polymorphism is a challenge in order-sorted logic.
- Intensional logic is required.
- Implicit guarding makes it compact and elegant.
- Complex well-typing relation (future work).

Order-sorted Intensional logic – Conclusion

- Expressing subtyping polymorphism is a challenge in order-sorted logic.
- Intensional logic is required.
- Implicit guarding makes it compact and elegant.
- Complex well-typing relation (future work).

Order-sorted Intensional logic – Conclusion

- Expressing subtyping polymorphism is a challenge in order-sorted logic.
- Intensional logic is required.
- Implicit guarding makes it compact and elegant.
- Complex well-typing relation (future work).

? Ambiguities in Knowledge Representation

~~Partial functions~~

~~Decision modeling~~

~~Order-sorted Intensional logic~~


Summary

Summary

We covered:

- Basics of the KR paradigm.
- First-order logic with inductive definitions as a modeling language (with example).
- Problems with partial functions.
- Role of epistemic logic in rule-based decision modeling languages.
- Expressive properties of order-sorted intensional logic.

Summary

We covered:

- Basics of the KR paradigm.
- First-order logic with inductive definitions as a modeling language (with example).
- Problems with partial functions.
- Role of epistemic logic in rule-based decision modeling languages.
- Expressive properties of order-sorted intensional logic.

Summary

We covered:

- Basics of the KR paradigm.
- First-order logic with inductive definitions as a modeling language (with example).
- Problems with partial functions.
- Role of epistemic logic in rule-based decision modeling languages.
- Expressive properties of order-sorted intensional logic.

Summary

We covered:

- Basics of the KR paradigm.
- First-order logic with inductive definitions as a modeling language (with example).
- Problems with partial functions.
- Role of epistemic logic in rule-based decision modeling languages.
- Expressive properties of order-sorted intensional logic.

Summary

We covered:

- Basics of the KR paradigm.
- First-order logic with inductive definitions as a modeling language (with example).
- Problems with partial functions.
- Role of epistemic logic in rule-based decision modeling languages.
- Expressive properties of order-sorted intensional logic.

Summary

In Knowledge Representation it is very important to get the models right!

Summary

In Knowledge Representation it is very important to get the models right!

Term "model" is overloaded; what I mean:

1. To make the formalization (or modeling) of the particular domain correspond as much as possible to the informal understanding of that domain.
2. To ensure that the formalization yields correct models (i,e., possible worlds of the theory according to the model semantics are isomorphic to the possible state of affairs of the real world.)

Summary

In Knowledge Representation it is very important to get the models right!

Term "model" is overloaded; what I mean:

1. To make the formalization (or modeling) of the particular domain correspond as much as possible to the informal understanding of that domain.
2. To ensure that the formalization yields correct models (i,e., possible worlds of the theory according to the model semantics are isomorphic to the possible state of affairs of the real world.)

Summary

In Knowledge Representation it is very important to get the models right!

Term "model" is overloaded; what I mean:

1. To make the formalization (or modeling) of the particular domain correspond as much as possible to the informal understanding of that domain.
2. To ensure that the formalization yields correct models (i,e., possible worlds of the theory according to the model semantics are isomorphic to the possible state of affairs of the real world.)

Thank you for your attention

https://djordje.rs/

Đorđe Marković
dorde.markovic@kuleuven.be
markovic@djordje.rs