Link to this presentation – A Prudent Logic of Partial Functions

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

A Prudent Logic of Partial Functions

Đorđe Marković

November 12, 2024
KU Leuven, Leuven, Belgium

Content

- Partial functions?
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

Content

- Partial functions?
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

Content

- ÷ Partial functions?
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

Content

- Partial functions?
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

Content

- Partial functions?
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

÷ Partial functions?

÷ Partial functions?

Some examples:

- Subtraction in Natural numbers (i.e., $5 - 10$).
- Division in Natural and Real numbers (i.e., $5/0$).
- The present king of France is bald.
- Next element of a list.

÷ Partial functions?

Some examples:

- Subtraction in Natural numbers (i.e., $5 - 10$).
- Division in Natural and Real numbers (i.e., $5/0$).
- The present king of France is bald.
- Next element of a list.

÷ Partial functions?

Some examples:

- Subtraction in Natural numbers (i.e., $5 - 10$).
- Division in Natural and Real numbers (i.e., $5/0$).
- The present king of France is bald.
- Next element of a list.

÷ Partial functions?

Some examples:

- Subtraction in Natural numbers (i.e., $5 - 10$).
- Division in Natural and Real numbers (i.e., $5/0$).
- The present king of France is bald.
- Next element of a list.

÷ Partial functions?

Some examples:

- Subtraction in Natural numbers (i.e., $5 - 10$).
- Division in Natural and Real numbers (i.e., $5/0$).
- The present king of France is bald.
- Next element of a list.

÷ Partial functions?

- What is the issue with:

The present king of France is bald.

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions?

- What is the issue with:

The present king of France is bald.

- Who thinks the statement is **true**? ✋
- Who thinks the statement is **false**?
- Who thinks the statement is **neither** true nor false?

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions?

- What is the issue with:

The present king of France is bald.

- Who thinks the statement is **true**?
- Who thinks the statement is **false**? ✋
- Who thinks the statement is **neither** true nor false?

On denoting. Bertrand Russell. Mind, 1905

➗ Partial functions?

- What is the issue with:

The present king of France is bald.

- Who thinks the statement is **true**?
- Who thinks the statement is **false**?
- Who thinks the statement is **neither** true nor false? ✋

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions?

- 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!
  Senseless sentences have no meaning. Is the sentence above senseless?

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions?

- 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!

  Senseless sentences have no meaning. Is the sentence above senseless?

On denoting. Bertrand Russell. Mind, 1905

÷ Partial functions?

- 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!

Senseless sentences have no meaning. Is the sentence above senseless?

On denoting. Bertrand Russell. Mind, 1905
On referring. Peter Frederick Strawson. Mind, 1950

÷ Partial functions?

- 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!
Senseless sentences have no meaning. Is the sentence above senseless?

On denoting. Bertrand Russell. Mind, 1905
On referring. Peter Frederick Strawson. Mind, 1950

÷ Partial functions?

How about this statement:

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

Is it true of false (or neither)?

÷ Partial functions?

Let's give it another try!

Why we are inclined to say that the following statement is false?

The present king of France is bald.

- Russell explained by giving this sentence the following meaning:

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

Why we are inclined to say that the following statement is true?

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

- I believe because we interpret the sentence as:

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

÷ Partial functions?

Let's give it another try!

Why we are inclined to say that the following statement is false?

The present king of France is bald.

- Russell explained by giving this sentence the following meaning:

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

Why we are inclined to say that the following statement is true?

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

- I believe because we interpret the sentence as:

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

÷ Partial functions?

Let's give it another try!

Why we are inclined to say that the following statement is false?

The present king of France is bald.

- Russell explained by giving this sentence the following meaning:

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

Why we are inclined to say that the following statement is true?

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

- I believe because we interpret the sentence as:

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

÷ Partial functions?

Let's give it another try!

Why we are inclined to say that the following statement is false?

The present king of France is bald.

- Russell explained by giving this sentence the following meaning:

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

Why we are inclined to say that the following statement is true?

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

- I believe because we interpret the sentence as:

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

÷ Partial functions?

Wait a moment!

Why then we do not interpret:

The present king of France is bald.

as:

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

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

÷ Partial functions?

Wait a moment!

Why then we do not interpret:

The present king of France is bald.

as:

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

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

Content

- ÷ ~~Partial functions?~~
- Knowledge representation with partial functions!
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

Knowledge representation with partial functions!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1 :$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : Vertex(x) \Rightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Formally, no problem with partial functions!
- However, there is a semantic mismatch!

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad \textit{partial}\ colOf/1 :$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : colOf(x) = y \Rightarrow Vertex(x) \land Color(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Problematic statement!
- How to make sure the theory is well-defined (i.e., true or false in every structure)?

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad \textit{partial}\ colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : colOf(x) = y \Rightarrow Vertex(x) \land Color(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Problematic statement!
- How to make sure the theory is well-defined (i.e., true or false in every structure)?

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad \textit{partial}\ colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : colOf(x) = y \Rightarrow Vertex(x) \land Color(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Problematic statement!
- How to make sure the theory is well-defined (i.e., true or false in every structure)?

Knowledge representation with partial functions!

Consider the graph coloring problem modeled in first-order logic:

- Vocabulary:

$$Vertex/1 \qquad Color/1 \qquad Graph/2 \qquad \textit{partial}\ colOf/1:$$

- Theory:

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : colOf(x) = y \Rightarrow Vertex(x) \land Color(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

- Problematic statement!
- How to make sure the theory is well-defined (i.e., true or false in every structure)?

Knowledge representation with partial functions!

How to make sure the theory is well-defined (i.e., true or false in every structure)?

- “Type Correctness Condition” - Translate theory into another one and check its validity!
- TCC translation enjoies the well-definedness property!
- Furthermore, if TCC translation is valid, partial functions can be arbitrarily extended to a total while preserving the meaning of the original theory.
- However, validity checking is undecidable!

A practical approach to partial functions in CVC lite. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.L. Electronic Notes in Theoretical Computer Science, 2005

Knowledge representation with partial functions!

How to make sure the theory is well-defined (i.e., true or false in every structure)?

- "Type Correctness Condition" - Translate theory into another one and check its validity!
- TCC translation enjoies the well-definedness property!
- Furthermore, if TCC translation is valid, partial functions can be arbitrarily extended to a total while preserving the meaning of the original theory.
- However, validity checking is undecidable!

A practical approach to partial functions in CVC lite. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.L. Electronic Notes in Theoretical Computer Science, 2005

Knowledge representation with partial functions!

How to make sure the theory is well-defined (i.e., true or false in every structure)?

- "Type Correctness Condition" - Translate theory into another one and check its validity!
- TCC translation enjoies the well-definedness property!
- Furthermore, if TCC translation is valid, partial functions can be arbitrarily extended to a total while preserving the meaning of the original theory.
- However, validity checking is undecidable!

A practical approach to partial functions in CVC lite. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.L. Electronic Notes in Theoretical Computer Science, 2005

Knowledge representation with partial functions!

How to make sure the theory is well-defined (i.e., true or false in every structure)?

- "Type Correctness Condition" - Translate theory into another one and check its validity!
- TCC translation enjoies the well-definedness property!
- Furthermore, if TCC translation is valid, partial functions can be arbitrarily extended to a total while preserving the meaning of the original theory.
- However, validity checking is undecidable!

A practical approach to partial functions in CVC lite. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.L. Electronic Notes in Theoretical Computer Science, 2005

Knowledge representation with partial functions!

How to make sure the theory is well-defined (i.e., true or false in every structure)?

- ~~"Type Correctness Condition" - Translate theory into another one and check its validity!~~
- TCC translation enjoies the well-definedness property!
- Furthermore, if TCC translation is valid, partial functions can be arbitrarily extended to a total while preserving the meaning of the original theory.
- However, validity checking is undecidable!

A practical approach to partial functions in CVC lite. Berezin, S., Barrett, C., Shikanian, I., Chechik, M., Gurfinkel, A., Dill, D.L. Electronic Notes in Theoretical Computer Science, 2005

Content

- ~~Partial functions?~~
- ~~Knowledge representation with partial functions!~~
- A Prudent Logic of Partial Functions
- Recursive definitions
- Conclusion

A Prudent Logic of Partial Functions

A Prudent Logic of Partial Functions

What we aim to improve?

- Checking well-definedness should be (efficiently) decidable.
- Improve error reporting!

How?

- Make well-definedness syntactic property of the language (guarded partial functions)!

A Prudent Logic of Partial Functions

What we aim to improve?

- Checking well-definedness should be (efficiently) decidable.
- Improve error reporting!

How?

- Make well-definedness syntactic property of the language (guarded partial functions)!

A Prudent Logic of Partial Functions

What we aim to improve?

- Checking well-definedness should be (efficiently) decidable.
- Improve error reporting!

How?

- Make well-definedness syntactic property of the language (guarded partial functions)!

A Prudent Logic of Partial Functions

What we aim to improve?

- Checking well-definedness should be (efficiently) decidable.
- Improve error reporting!

How?

- Make well-definedness syntactic property of the language (guarded partial functions)!

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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$

- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:
$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$
- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$

- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$

- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$

- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$

- 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 then $[\delta_f(t_1, \ldots, t_n)]^{\mathfrak{A}}$ is **true**.
- Defining domain example:
$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x).$$
- 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:

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

- 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:

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

- 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:

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

- 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:

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

- 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:

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

- 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

Guarding is not easy!

- Consider the following example:

  My partner's mother is a doctor.

- In first-order logic:

$$Doctor(mother(mp))$$

- Guarded:

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

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:

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

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:

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

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:

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

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:

$$\begin{aligned}&\textbf{if } \delta_{mp}() \textbf{ then}\\ &\qquad \textbf{if } \delta_{mother}(mp) \textbf{ then } Doctor(mother(mp)) \textbf{ else } \mathit{false} \textbf{ fi}\\ &\textbf{else } \mathit{false} \textbf{ fi}.\end{aligned}$$

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:

$$\begin{aligned}&\textbf{if } \delta_{mp}() \textbf{ then}\\ &\qquad \textbf{if } \delta_{mother}(mp) \textbf{ then } Doctor(mother(mp)) \textbf{ else } \mathit{false} \textbf{ fi}\\ &\textbf{else } \mathit{false} \textbf{ fi}.\end{aligned}$$

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

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

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

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

Example: My partner's mother is a doctor.

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

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

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

$$\phi \overrightarrow{\Rightarrow} \psi \doteq \textbf{if } \phi \textbf{ then } \psi \textbf{ else } \textit{true} \textbf{ fi}$$

Implicit guards (annotations):

My partner's mother is a doctor.

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

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

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

Guarded Partial Function Logic – Conveniences

Conjunctive and implicative guards:

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

Implicit guards (annotations):

My partner's mother is a doctor.

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

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

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

Guarded Partial Function Logic – Conveniences

Term guards:

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

Example: My partner's mother is a doctor.

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

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: My partner's mother is a doctor.

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

Formally defined as:

$$\Delta(x) \doteq true$$

$$\Delta(f(s_1, \dots, s_n)) \doteq \Delta(s_1) \overrightarrow{\wedge} \dots \overrightarrow{\wedge} \Delta(s_n) \overrightarrow{\wedge} \delta_f(s_1, \dots, s_n)$$

Guarded Partial Function Logic – Conveniences

Term guards:

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

Example: My partner's mother is a doctor.

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

Formally defined as:

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

Guarded Partial Function Logic - Main results

- We use strong Kleene three-valued semantics.
- A formula is well-guarded if all its partial function terms are guarded.
- 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** - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- **Theorem 3** - 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.
- A formula is well-guarded if all its partial function terms are guarded.
- 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** - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- **Theorem 3** - 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.
- A formula is well-guarded if all its partial function terms are guarded.
- 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 - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- Theorem 3 - 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.
- A formula is well-guarded if all its partial function terms are guarded.
- 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** - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- **Theorem 3** - 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.
- A formula is well-guarded if all its partial function terms are guarded.
- 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** - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- **Theorem 3** - 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.
- A formula is well-guarded if all its partial function terms are guarded.
- 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** - If a formula is well-guarded, its value under the three-valued and two-valued semantics coincide (where the three-valued structure is arbitrarily extended).
- **Theorem 3** - 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

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \overset{\rightarrow}{\Rightarrow} Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \overset{\rightarrow}{\Rrightarrow} Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$

$$\forall x : \delta_{colOf}(x) \overset{\delta}{\Rrightarrow} Color(colOf(x))$$

$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$

$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \Rrightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \Rrightarrow Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \overset{\delta}{\Rightarrow} Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

Not all formulas in this theory are well-guarded!

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \mathrel{\overset{\rightarrow}{\Rightarrow}} Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

Not all formulas in this theory are well-guarded!

However, the theory is well-defined!

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \overrightarrow{\Rightarrow} Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \land Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

Not all formulas in this theory are well-guarded!

However, the theory is well-defined!

A Prudent Logic of Partial Functions

What is the price? Recall the graph coloring example (slightly rewritten):

$$\forall x : \delta_{colOf}(x) \Leftrightarrow Vertex(x)$$
$$\forall x : \delta_{colOf}(x) \rightrightarrows Color(colOf(x))$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow Vertex(x) \wedge Vertex(y)$$
$$\forall x : \forall y : Graph(x, y) \Rightarrow \neg(colOf(x) = colOf(y))$$

Not all formulas in this theory are well-guarded!

However, the theory is well-defined!

Indirect Guarding of Partial Functions

How to mitigate this issue:

- Extend guarding relation with Generalized Modus Ponens.
- Preserve the polynomial complexity!
- Presented results extend to the generalized guarding.
- There are well-defined theories that are not well-guarded, but the solution is promising for KR purposes.

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.

Indirect Guarding of Partial Functions

How to mitigate this issue:

- Extend guarding relation with Generalized Modus Ponens.
- Preserve the polynomial complexity!
- Presented results extend to the generalized guarding.
- There are well-defined theories that are not well-guarded, but the solution is promising for KR purposes.

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.

Indirect Guarding of Partial Functions

How to mitigate this issue:

- Extend guarding relation with Generalized Modus Ponens.
- Preserve the polynomial complexity!
- Presented results extend to the generalized guarding.
- There are well-defined theories that are not well-guarded, but the solution is promising for KR purposes.

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.

Indirect Guarding of Partial Functions

How to mitigate this issue:

- Extend guarding relation with Generalized Modus Ponens.
- Preserve the polynomial complexity!
- Presented results extend to the generalized guarding.
- There are well-defined theories that are not well-guarded, but the solution is promising for KR purposes.

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.

Indirect Guarding of Partial Functions

How to mitigate this issue:

- Extend guarding relation with Generalized Modus Ponens.
- Preserve the polynomial complexity!
- Presented results extend to the generalized guarding.
- There are well-defined theories that are not well-guarded, but the solution is promising for KR purposes.

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.

Content

- ~~Partial functions?~~
- ~~Knowledge representation with partial functions!~~
- ~~A Prudent Logic of Partial Functions~~
- Recursive definitions
- Conclusion

Recursive definitions

Recursive definitions

First-order logic with inductive definitions:

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

Extending Classical Logic with Inductive Definitions. Marc Denecker. Lecture Notes in Computer Science 2000

Recursive definitions

First-order logic with inductive definitions:

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

Extending Classical Logic with Inductive Definitions. Marc Denecker. Lecture Notes in Computer Science 2000

Recursive definitions

First-order logic with inductive definitions:

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

Extending Classical Logic with Inductive Definitions. Marc Denecker. Lecture Notes in Computer Science 2000

Recursive definitions

First-order logic with inductive definitions:

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

What if some of the parameters is partial function?

Extending Classical Logic with Inductive Definitions. Marc Denecker. Lecture Notes in Computer Science 2000

Recursive definitions

First-order logic with inductive definitions:

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

What if some of the parameters is partial function?

What if the defined concept is a partial function (i.e., recursive definition)?

Extending Classical Logic with Inductive Definitions. Marc Denecker. Lecture Notes in Computer Science 2000

Recursive definitions

Examples of recursive definitions:

- Spouse function:

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

- Favorite movie:

$$\left\{ \begin{array}{l} \forall p, m : favorite(p) = m \leftarrow \exists s : Score(p, m, s) \land \\ \qquad\qquad \neg\exists m', s' : m' \neq m \land Score(p, m', s') \land s' \geq s \end{array} \right\}$$

Recursive definitions

Examples of recursive definitions:

- Spouse function:

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

- Favorite movie:

$$\left\{\begin{array}{r} \forall p, m : favorite(p) = m \leftarrow \exists s : Score(p, m, s) \wedge \\ \neg \exists m', s' : m' \neq m \wedge Score(p, m', s') \wedge s' \geq s \end{array}\right\}$$

Recursive definitions

Examples of recursive definitions:

- Spouse function:

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

- Favorite movie:

$$\left\{ \begin{array}{r} \forall p, m : favorite(p) = m \leftarrow \exists s : Score(p, m, s) \wedge \\ \neg \exists m', s' : m' \neq m \wedge Score(p, m', s') \wedge s' \geq s \end{array} \right\}$$

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

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

- Body of a rule is well-guarded!

- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

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

- Body of a rule is well-guarded!

- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

$$\forall \bar{x} : P(\bar{t}) \leftarrow \Delta(\bar{t}) \land \phi$$
$$\forall \bar{x} : f(\bar{t}) = k \leftarrow \Delta(\bar{t}) \land \Delta(k) \land \phi$$

- Body of a rule is well-guarded!

- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

$$\forall \bar{x} : P(\bar{t}) \leftarrow \Delta(\bar{t}) \land \phi$$
$$\forall \bar{x} : f(\bar{t}) = k \leftarrow \Delta(\bar{t}) \land \Delta(k) \land \phi$$

- Body of a rule is well-guarded!
- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

$$\forall \bar{x} : P(\bar{t}) \leftarrow \Delta(\bar{t}) \land \phi$$
$$\forall \bar{x} : f(\bar{t}) = k \leftarrow \Delta(\bar{t}) \land \Delta(k) \land \phi$$

- Body of a rule is well-guarded!

- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions

How to integrate recursive definition with Logic of Partial Functions in a prudent way?

- All terms in the head are guarded (except the defined term)!

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

- Body of a rule is well-guarded!

- With these constraints we can reduce definitions of function to definitions of predicates!

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.

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n,0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n,0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \wedge m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n, 0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n, 0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \wedge m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n, 0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n, 0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \land m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n, 0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n, 0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \wedge m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n, 0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n, 0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \wedge m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l}
\forall n : plus(n, 0) = n \\
\forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\
\forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\
\forall n : times(n, 0) = 0 \\
\forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\
\forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \wedge m \neq 0
\end{array}\right\}$$

Recursive definitions - Example

The four elementary arithmetic operations can be defined mutually in FO(RD), in terms of the successor function $s/1$ and the constant 0:

$$\left\{\begin{array}{l} \forall n : plus(n, 0) = n \\ \forall n, m : plus(n, s(m)) = s(plus(n, m)) \leftarrow \Delta(s(plus(n, m))) \\ \forall n, m, p : minus(n, m) = p \leftarrow [\![n = plus(m, p)]\!] \\ \forall n : times(n, 0) = 0 \\ \forall n, m : times(n, s(m)) = plus(times(n, m), n) \leftarrow \Delta(plus(times(n, m), n)) \\ \forall n, m, p : div(n, m) = p \leftarrow [\![n = times(m, p)]\!] \land m \neq 0 \end{array}\right\}$$

Content

- ~~Partial functions?~~
- ~~Knowledge representation with partial functions!~~
- ~~A Prudent Logic of Partial Functions~~
- ~~Recursive definitions~~
- Conclusion

Conclusion

Conclusion

- Conservative extension of first-order logic with partial functions.
- Strict syntax (based on simple conditional guarding).
- Nice properties (well-definedness, semantic value preserved).
- Language conveniences introduced in terms of conditional guards.
- Extension with inductive definitions.

Conclusion

- Conservative extension of first-order logic with partial functions.
- Strict syntax (based on simple conditional guarding).
- Nice properties (well-definedness, semantic value preserved).
- Language conveniences introduced in terms of conditional guards.
- Extension with inductive definitions.

Conclusion

- Conservative extension of first-order logic with partial functions.
- Strict syntax (based on simple conditional guarding).
- Nice properties (well-definedness, semantic value preserved).
- Language conveniences introduced in terms of conditional guards.
- Extension with inductive definitions.

Conclusion

- Conservative extension of first-order logic with partial functions.
- Strict syntax (based on simple conditional guarding).
- Nice properties (well-definedness, semantic value preserved).
- Language conveniences introduced in terms of conditional guards.
- Extension with inductive definitions.

Conclusion

- Conservative extension of first-order logic with partial functions.
- Strict syntax (based on simple conditional guarding).
- Nice properties (well-definedness, semantic value preserved).
- Language conveniences introduced in terms of conditional guards.
- Extension with inductive definitions.

Thank you for your attention

A Prudent Logic of Partial Functions; Make sure your theory is always well-defined!

Đorđe Marković *dorde.markovic@kuleuven.be* https://djordje.rs/