Functions as Graphs, or Functions as Rules?

Einblicke in meine Bachelor-Arbeit

Übersicht

Organisatorisches

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:
    1. Sind meine Argumente jeweils schlüssig?

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:
    1. Sind meine Argumente jeweils schlüssig?
    2. Gibt es weitere Argumente gegen den Functional Egalitarian?

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:
    1. Sind meine Argumente jeweils schlüssig?
    2. Gibt es weitere Argumente gegen den Functional Egalitarian?
    3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:
    1. Sind meine Argumente jeweils schlüssig?
    2. Gibt es weitere Argumente gegen den Functional Egalitarian?
    3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?
    4. An welchen Stellen drücke ich mich unklar aus?

Organisatorisches

1. Der reine Vortrag wird voraussichtlich 50 Minuten dauern.
2. Bitte stellen Sie Verständnisfragen unmittelbar – auch während des Vortrags.
3. Am Ende des Vortrags folgt eine Diskussionsrunde. Bitte haben Sie deshalb während des Vortrags die folgenden Fragen im Kopf:
    1. Sind meine Argumente jeweils schlüssig?
    2. Gibt es weitere Argumente gegen den Functional Egalitarian?
    3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?
    4. An welchen Stellen drücke ich mich unklar aus?
    5. Gibt es nicht vielleicht doch ein Schlupfloch für den Functional Egalitarian?

Functional Egalitarianism

Functions-as-Rules

(Functions-as-Rules) Eine Funktion ist eine Regel, die angibt, was man mit einer festgelegten Anzahl von Gegenständen (Inputs) tun soll.

Regeln lassen sich auf zwei Weisen schreiben:

1. Mithilfe des Zuweisungsoperators $\mapsto$, wie in $\sigma : x \mapsto x + 1$,

Functions-as-Rules

(Functions-as-Rules) Eine Funktion ist eine Regel, die angibt, was man mit einer festgelegten Anzahl von Gegenständen (Inputs) tun soll.

Regeln lassen sich auf zwei Weisen schreiben:

1. Mithilfe des Zuweisungsoperators $\mapsto$, wie in $\sigma : \mathrm{x} \mapsto \mathrm{x} + 1$,
2. Mithilfe von Fallunterscheidungen, wie in

Functions-as-Rules

(Functions-as-Rules) Eine Funktion ist eine Regel, die angibt, was man mit einer festgelegten Anzahl von Gegenständen (Inputs) tun soll.

Regeln lassen sich auf zwei Weisen schreiben:

1. Mithilfe des Zuweisungsoperators $\mapsto$, wie in $\sigma : \mathrm{x} \mapsto \mathrm{x} + 1$,
2. Mithilfe von Fallunterscheidungen, wie in

Functions-as-Rules

(Functions-as-Rules) Eine Funktion ist eine Regel, die angibt, was man mit einer festgelegten Anzahl von Gegenständen (Inputs) tun soll.

Regeln lassen sich auf zwei Weisen schreiben:

1. Mithilfe des Zuweisungsoperators $\mapsto$, wie in $\sigma : x \mapsto x + 1$,
2. Mithilfe von Fallunterscheidungen, wie in

$$d(x) = \begin{cases} 1, & \text{falls } x \in \mathbb{Q} \\ 0, & \text{falls } x \in \mathbb{R} \setminus \mathbb{Q}. \end{cases}$$

Functions-as-Graphs

(Functions-as-Graphs) Eine Funktion ist eine Menge von Argument-Wert-Paaren, sodass jedem Argument genau ein Wert zugeordnet ist.

Graphen lassen sich auf zwei Weisen angeben:

1. Durch explizite Aufzählung der Argument-Wert-Paare, wie in $\sigma = \{\langle 0, 1\rangle, \langle 1, 2\rangle, ...\}$,

Functions-as-Graphs

(Functions-as-Graphs) Eine Funktion ist eine Menge von Argument-Wert-Paaren, sodass jedem Argument genau ein Wert zugeordnet ist.

Graphen lassen sich auf zwei Weisen angeben:

1. Durch explizite Aufzählung der Argument-Wert-Paare, wie in $\sigma = \{\langle 0, 1\rangle, \langle 1, 2\rangle, ...\}$,

2. Durch eingeschränkte Komprehension, wie in $\sigma = \{\langle x, y\rangle \in \mathbb{N} \times \mathbb{N} \mid y = x + 1\}$.

Was ist Functional Egalitarianism?

Offensichtlich behaupten (Functions-as-Rules) und (Functions-as-Graphs) jeweils etwas Anderes über das Wesen von Funktionen.

Was ist Functional Egalitarianism?

Offensichtlich behaupten (Functions-as-Rules) und (Functions-as-Graphs) jeweils etwas Anderes über das Wesen von Funktionen.

Trotzdem könnte man denken, dass beide Auffassungen dieselben Funktionen definieren und die Debatte deshalb als unnötig empfinden. Nennen wir diese Ansicht functional egalitarianism und einen Vertreter davon einen functional egalitarian.

Die Behauptungen eines Functional Egalitarians

Ein functional egalitarian würde die Konjunktion dieser beiden Auffassungen vertreten:

(Rules-as-Graphs)	Für jede Regel lässt sich ein ihr entsprechender Graph konstruieren.
(Graphs-as-Rules)	Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

Ziel des Vortrags

Ziel dieses Vortrags ist zu zeigen, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. Spezifisch werde ich dafür argumentieren, dass ...

1. (Rules-as-Graphs) zwar von einem intuitiv zugänglichen Prinzip gestützt würde, dieses Prinzip aber

Ziel des Vortrags

Ziel dieses Vortrags ist zu zeigen, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. Spezifisch werde ich dafür argumentieren, dass ...

1. (Rules-as-Graphs) zwar von einem intuitiv zugänglichen Prinzip gestützt würde, dieses Prinzip aber
    1. zum einen schlichtweg falsch ist und

Ziel des Vortrags

Ziel dieses Vortrags ist zu zeigen, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. Spezifisch werde ich dafür argumentieren, dass ...

1. (Rules-as-Graphs) zwar von einem intuitiv zugänglichen Prinzip gestützt würde, dieses Prinzip aber
    1. zum einen schlichtweg falsch ist und
    2. zum anderen weitere, unplausible Prämissen benötigt, um (Rules-as-Graphs) zu implizieren.

Ziel des Vortrags

Ziel dieses Vortrags ist zu zeigen, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. Spezifisch werde ich dafür argumentieren, dass ...

1. (Rules-as-Graphs) zwar von einem intuitiv zugänglichen Prinzip gestützt würde, dieses Prinzip aber
    1. zum einen schlichtweg falsch ist und
    2. zum anderen weitere, unplausible Prämissen benötigt, um (Rules-as-Graphs) zu implizieren.

2. es für (Graphs-as-Rules) zwar einen Trick gibt, für jeden *endlichen* Graphen eine Regel zu finden, dieser für *unendliche* Graphen aufgrund des Auswahlaxioms allerdings nicht funktioniert.

Probleme mit (Rules-as-Graphs)

Das Prinzip (Rule-to-Graph)

Es scheint ein intuitives Prinzip zu geben, das (Rules-as-Graphs) stützt:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Idee hinter (Rule-to-Graph) ist diese:

1. Mit Input und Regel gelangt man zum Output.

Das Prinzip (Rule-to-Graph)

Es scheint ein intuitives Prinzip zu geben, das (Rules-as-Graphs) stützt:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Idee hinter (Rule-to-Graph) ist diese:

1. Mit Input und Regel gelangt man zum Output.
2. Mit Input und Output kann man einen Input-Output-Tupel erzeugen.

Das Prinzip (Rule-to-Graph)

Es scheint ein intuitives Prinzip zu geben, das (Rules-as-Graphs) stützt:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Idee hinter (Rule-to-Graph) ist diese:

1. Mit Input und Regel gelangt man zum Output.
2. Mit Input und Output kann man einen Input-Output-Tupel erzeugen.
3. Macht man das für jeden Input, erhält man den Graphen der Regel.

Das Prinzip (Rule-to-Graph)

Es scheint ein intuitives Prinzip zu geben, das (Rules-as-Graphs) stützt:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Idee hinter (Rule-to-Graph) ist diese:

1. Mit Input und Regel gelangt man zum Output.
2. Mit Input und Output kann man einen Input-Output-Tupel erzeugen.
3. Macht man das für jeden Input, erhält man den Graphen der Regel.

Das Prinzip (Rule-to-Graph)

Es scheint ein intuitives Prinzip zu geben, das (Rules-as-Graphs) stützt:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Idee hinter (Rule-to-Graph) ist diese:

1. Mit Input und Regel gelangt man zum Output.
2. Mit Input und Output kann man einen Input-Output-Tupel erzeugen.
3. Macht man das für jeden Input, erhält man den Graphen der Regel.

Beispielsweise kann man mit der Regel $\mathrm{x} \mapsto 2\mathrm{x}$ sowie den Argumenten 2 und 8 den Graphen $\{\langle 2, 2 \times 2\rangle, \langle 8, 2 \times 8\rangle\}$ konstruieren.

Die Schwachstellen von (Rule-to-Graph)

(Rule-to-Graph) ist eine sehr starke Aussage, weil sie gleich zwei Allquantifikationen enthält:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Die Schwachstellen von (Rule-to-Graph)

(Rule-to-Graph) ist eine sehr starke Aussage, weil sie gleich zwei Allquantifikationen enthält:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Im Gegensatz zu Mengen gibt es keine Beschränkungen auf Selbstreferenz bei Regeln.

Die Schwachstellen von (Rule-to-Graph)

(Rule-to-Graph) ist eine sehr starke Aussage, weil sie gleich zwei Allquantifikationen enthält:

(Rule-to-Graph) Für jede Regel R and beliebige Inputs I lässt sich der Graph von R für I konstruieren.

Im Gegensatz zu Mengen gibt es keine Beschränkungen auf Selbstreferenz bei Regeln.

Im Folgenden werde ich diese Tatsache ausnutzen, um zu zeigen, dass es Gegenbeispiele für beide Quantifikationen gibt.

(Rule-to-Graph): Wirklich jede Regel?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede selbstreferierende Regel und jeden Input ein Graph konstruieren.

(Rule-to-Graph): Wirklich jede Regel?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede selbstreferierende Regel und jeden Input ein Graph konstruieren.

Zunächst lässt sich festhalten, dass es selbstreferierende Regeln gibt, für die man einen Graph konstruieren kann, nämlich die rekursiven Regeln. Ein Beispiel:

$$\mathrm{x}! = \begin{cases} 1, & \text{falls } \mathrm{x} = 0 \\ \mathrm{x} \times (\mathrm{x}-1)! & \text{sonst} \end{cases}$$

(Rule-to-Graph): Wirklich jede Regel?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede selbstreferierende Regel und jeden Input ein Graph konstruieren.

Zunächst lässt sich festhalten, dass es selbstreferierende Regeln gibt, für die man einen Graph konstruieren kann, nämlich die rekursiven Regeln. Ein Beispiel:

$$\mathrm{x!} = \begin{cases} 1, & \text{falls } \mathrm{x} = 0 \\ \mathrm{x} \times (\mathrm{x}-1)! & \text{sonst} \end{cases}$$

Zwar kommt „!“ in der Funktionsdefinition von ! vor, aber es lässt sich auflösen:

$$\begin{aligned} \{\langle 0,1\rangle, \langle 1, 1\times 0!\rangle, \langle 2, 2\times 1!\rangle, ...\} &= \{\langle 0,1\rangle, \langle 1, 1\times 1\rangle, \langle 2, 2\times 1\times 1\rangle, ...\} \\ &= \{\langle 0,1\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle, ...\} \end{aligned}$$

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

Nennen wir diejenigen Regeln, die in ihrer Definition auf sich selbst Bezug nehmen, reokkursive Regeln. Diese Regeln sind die potentiellen Gegenbeispiele für die Allquantifikation über Regeln.

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

Nennen wir diejenigen Regeln, die in ihrer Definition auf sich selbst Bezug nehmen, reokkursive Regeln. Diese Regeln sind die potentiellen Gegenbeispiele für die Allquantifikation über Regeln.

Die reokkursiven Regeln lassen sich wie folgt einteilen:

1. Die rekursiven Regeln, deren Wiederholung aufhört, weil sie auf einen Basisfall zurückläuft.

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

Nennen wir diejenigen Regeln, die in ihrer Definition auf sich selbst Bezug nehmen, reokkursive Regeln. Diese Regeln sind die potentiellen Gegenbeispiele für die Allquantifikation über Regeln.

Die reokkursiven Regeln lassen sich wie folgt einteilen:

1. Die rekursiven Regeln, deren Wiederholung aufhört, weil sie auf einen Basisfall zurückläuft.
2. Die prokursiven Regeln, deren Wiederholung nie aufhört, weil sie lediglich vorläuft

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

Nennen wir diejenigen Regeln, die in ihrer Definition auf sich selbst Bezug nehmen, reokkursive Regeln. Diese Regeln sind die potentiellen Gegenbeispiele für die Allquantifikation über Regeln.

Die reokkursiven Regeln lassen sich wie folgt einteilen:

1. Die rekursiven Regeln, deren Wiederholung aufhört, weil sie auf einen Basisfall zurückläuft.
2. Die prokursiven Regeln, deren Wiederholung nie aufhört, weil sie lediglich vorläuft.

(Rule-to-Graph): Wirklich jede Regel?

Dennoch: Nur, weil es einige selbstreferierende Regeln gibt, für die sich ein Graph konstruieren lässt, heißt das nicht, dass das für alle gilt.

Nennen wir diejenigen Regeln, die in ihrer Definition auf sich selbst Bezug nehmen, reokkursive Regeln. Diese Regeln sind die potentiellen Gegenbeispiele für die Allquantifikation über Regeln.

Die reokkursiven Regeln lassen sich wie folgt einteilen:

1. Die rekursiven Regeln, deren Wiederholung aufhört, weil sie auf einen Basisfall zurückläuft.
2. Die prokursiven Regeln, deren Wiederholung nie aufhört, weil sie lediglich vorläuft.

Zwar haben wir gesehen, dass es mit rekursiven Regeln keine Probleme gibt. Bei prokursiven Regeln gibt es aber Probleme!

(Rule-to-Graph): Wirklich jede Regel?

Schauen wir uns ein Beispiel einer solchen prokursiven Regel an:

$$\mathrm{nobase}(\mathrm{x}) = \begin{cases} \mathrm{nobase}(\mathrm{x}+1), & \text{falls x gerade} \\ \mathrm{nobase}(\mathrm{x}+3), & \text{falls x ungerade} \end{cases}$$

(Rule-to-Graph): Wirklich jede Regel?

Schauen wir uns ein Beispiel einer solchen prokursiven Regel an:

$$\text{nobase}(\text{x}) = \begin{cases} \text{nobase}(\text{x}+1), & \text{falls x gerade} \\ \text{nobase}(\text{x}+3), & \text{falls x ungerade} \end{cases}$$

Offensichtlich addiert nobase Zahlen ad infinitum – es läuft immer vor, nie zurück, wie in

$$\text{nobase}(2) = \text{nobase}(2+1) = \text{nobase}(3) = \text{nobase}(3+3) = \text{nobase}(6) = \text{nobase}(6+1) = ...$$

(Rule-to-Graph): Wirklich jede Regel?

$$\text{nobase}(2) = \text{nobase}(2+1) = \text{nobase}(3) = \text{nobase}(3+3) = \text{nobase}(6) = \text{nobase}(6+1) = ...$$

Da nobase nie aufhört, wird nobase für keinen Input je einen Output haben. Aber gerade diesen Output benötigen wir, um einen Graph zu erzeugen. Also existiert kein Graph für nobase.

(Rule-to-Graph): Wirklich jede Regel?

$$\text{nobase}(2) = \text{nobase}(2+1) = \text{nobase}(3) = \text{nobase}(3+3) = \text{nobase}(6) = \text{nobase}(6+1) = ...$$

Da nobase nie aufhört, wird nobase für keinen Input je einen Output haben. Aber gerade diesen Output benötigen wir, um einen Graph zu erzeugen. Also existiert kein Graph für nobase.

Aber nobase ist eine Regel! Also gibt es mindestens eine Regel, sodass für egal welche Inputs kein Graph existiert. damit ist (Rule-to-Graph) widerlegt!

(Rule-to-Graph): Wirklich jeder Input?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede Regel und jeden Input ein Graph konstruieren, auch wenn der Input die Regel selbst ist.

(Rule-to-Graph): Wirklich jeder Input?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede Regel und jeden Input ein Graph konstruieren, auch wenn der Input die Regel selbst ist.

Betrachten wir dafür die Regel, die ihren Input auf sich selbst anwendet:

(Ω) $\quad \lambda x.\, xx$

(Rule-to-Graph): Wirklich jeder Input?

Wenn (Rule-to-Graph) wahr wäre, dann ließe sich für jede Regel und jeden Input ein Graph konstruieren, auch wenn der Input die Regel selbst ist.

Betrachten wir dafür die Regel, die ihren Input auf sich selbst anwendet:

$(\Omega) \qquad \lambda x.\, xx$

Wenn wir diese Regel auf sich selbst an, entsteht eine nicht endende Anwendungsschleife:

$(\Omega\Omega) \qquad (\lambda x.\, xx)(\lambda x.\, xx) \rightarrow_\beta (\lambda x.\, xx)(\lambda x.\, xx) \rightarrow_\beta \ldots$

(Rule-to-Graph): Wirklich jeder Input?

(ΩΩ) $(\lambda x.\, xx)(\lambda x.\, xx) \to_\beta (\lambda x.\, xx)(\lambda x.\, xx) \to_\beta \ldots$

Ähnlich wie bei nobase existiert kein Output für (ΩΩ). Also lässt sich auch hier kein Argument-Wert-Paar konstruieren, und damit auch kein Graph.

(Rule-to-Graph): Wirklich jeder Input?

$(\Omega\Omega)$ $\quad (\lambda x.\, xx)(\lambda x.\, xx) \to_\beta (\lambda x.\, xx)(\lambda x.\, xx) \to_\beta \ldots$

Ähnlich wie bei nobase existiert kein Output für $(\Omega\Omega)$. Also lässt sich auch hier kein Argument-Wert-Paar konstruieren, und damit auch kein Graph.

Das Beispiel von (Ω) unterscheidet sich vom nobase-Beispiel darin, dass einige Inputs für (Ω) einen Output haben, aber kein Input für nobase einen Output.

(Rule-to-Graph): Nötige Zusatzannahmen

Nehmen wir für einen Moment an, der functional egalitarian würde doch einen Ausweg finden, der ihm erlaubt, überzeugend für (Rules-as-Graphs) zu argumentieren. Wir zeigen nun, dass selbst das ihn nicht retten würde.

(Rule-to-Graph): Nötige Zusatzannahmen

Nehmen wir für einen Moment an, der functional egalitarian würde doch einen Ausweg finden, der ihm erlaubt, überzeugend für (Rules-as-Graphs) zu argumentieren. Wir zeigen nun, dass selbst das ihn nicht retten würde.

(Rule-to-Graph) impliziert (Rules-as-Graphs) nämlich nicht allein, sondern nur zusammen mit

(Rule-Inputs) Jede Regel ist mit einigen Inputs assoziiert.

(Rule-to-Graph): Nötige Zusatzannahmen

Nehmen wir für einen Moment an, der functional egalitarian würde doch einen Ausweg finden, der ihm erlaubt, überzeugend für (Rules-as-Graphs) zu argumentieren. Wir zeigen nun, dass selbst das ihn nicht retten würde.

(Rule-to-Graph) impliziert (Rules-as-Graphs) nämlich nicht allein, sondern nur zusammen mit

(Rule-Inputs) Jede Regel ist mit einigen Inputs assoziiert.

Gäbe es nämlich nur eine einzige Regel, von der wir nicht wüssten, auf welche Objekte sie angewendet werden soll, können wir auch kein einziges Input-Output-Paar erzeugen – und damit keinen Graphen, was (Rules-as-Graphs) widerspricht.

(Rule-to-Graph): Nötige Zusatzannahmen

Das Problem mit (Rule-Inputs) ist, dass nicht klar ist, was mit „assoziiert“ gemeint ist: Gehören die Inputs mit zur Regel? Falls ja, wäre $\mathrm{x} \mapsto 2\mathrm{x}$ auf 0 und 1 eine andere Regel als $\mathrm{x} \mapsto 2\mathrm{x}$ definiert auf 2 und 3? Ein functional egalitarian könnte antworten:

(Rule-to-Graph): Nötige Zusatzannahmen

Das Problem mit (Rule-Inputs) ist, dass nicht klar ist, was mit „assoziiert“ gemeint ist: Gehören die Inputs mit zur Regel? Falls ja, wäre $\mathrm{x} \mapsto 2\mathrm{x}$ auf 0 und 1 eine andere Regel als $\mathrm{x} \mapsto 2\mathrm{x}$ definiert auf 2 und 3? Ein functional egalitarian könnte antworten:

> Natürlich gehören die assoziierten Inputs nicht zur Regel. Die Regel ist $\mathrm{x} \mapsto 2\mathrm{x}$, aber man wendet sie eben auf verschiedene Inputs an. Wo ist das Problem mit „assoziiert“? Immer, wenn man eine Regel nutzt, sagt man, auf welche Inputs man sie anwendet. So einfach ist das.

(Rule-to-Graph): Nötige Zusatzannahmen

Problem: Wenn man behauptet, dass zu jeder Regel ein entsprechender Graph existiert, und Regeln separat von Inputs sieht, dann muss man für jede Regel *als solche* – ganz ohne Inputs – einen entsprechenden Graphen finden.

(Rule-to-Graph): Nötige Zusatzannahmen

Problem: Wenn man behauptet, dass zu jeder Regel ein entsprechender Graph existiert, und Regeln separat von Inputs sieht, dann muss man für jede Regel *als solche* – ganz ohne Inputs – einen entsprechenden Graphen finden.

Antwort vom functional egalitarian:

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

(Rule-to-Graph): Nötige Zusatzannahmen

Problem: Wenn man behauptet, dass zu jeder Regel ein entsprechender Graph existiert, und Regeln separat von Inputs sieht, dann muss man für jede Regel *als solche* – ganz ohne Inputs – einen entsprechenden Graphen finden.

Antwort vom functional egalitarian:

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Idee: Wenn wir den Graphen von R für *alle* sinnvollen Inputs definieren, dann sind alle Graphen für *einige* Inputs Teilmengen davon. In diesem Sinne decken wir also alle möglichen Graphen für R mit einer Menge von Inputs ab.

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.
3. Der Argumentbereich des Graphen einer Regel ist die Menge seiner Inputs.

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.
3. Der Argumentbereich des Graphen einer Regel ist die Menge seiner Inputs.
4. Daraus lässt sich schließen:

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.
3. Der Argumentbereich des Graphen einer Regel ist die Menge seiner Inputs.
4. Daraus lässt sich schließen:

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.
3. Der Argumentbereich des Graphen einer Regel ist die Menge seiner Inputs.
4. Daraus lässt sich schließen:

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

(Rule-to-Graph): Nötige Zusatzannahmen

(All Inputs) Der Graph, der einer Regel R entspricht, ist der für alle sinnvollen Inputs für R.

Problem:

1. (Rule-to-Graph) behauptet, dass für jede Regel ein entsprechender Graph existiert.
2. (All Inputs) behauptet, dass jede Regel mit all ihren sinnvollen Inputs assoziiert sind.
3. Der Argumentbereich des Graphen einer Regel ist die Menge seiner Inputs.
4. Daraus lässt sich schließen:

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

$\rightarrow$ Der Übergang von Regel zu Graph zieht die mengentheoretischen Restriktionen mit sich!

(Rule-to-Graph): Nötige Zusatzannahmen

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

(Rule-to-Graph): Nötige Zusatzannahmen

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

Diese Behauptung ist falsch. Gegenbeispiel: $x \mapsto x$. Hier gibt es gleich zwei Antinomien:

1. Offensichtlich ist jeder Input für die Identitätsregel sinnvoll – auch jede Menge. Also müsste die Allmenge Teilmenge des Argumentsbereichs von $x \mapsto x$ sein. Aber die Allmenge existiert nicht!

(Rule-to-Graph): Nötige Zusatzannahmen

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

Diese Behauptung ist falsch. Gegenbeispiel: $\mathrm{x} \mapsto \mathrm{x}$. Hier gibt es gleich zwei Antinomien:

1. Offensichtlich ist jeder Input für die Identitätsregel sinnvoll – auch jede Menge. Also müsste die Allmenge Teilmenge des Argumentsbereichs von $\mathrm{x} \mapsto \mathrm{x}$ sein. Aber die Allmenge existiert nicht!
2. Wendet man $\mathrm{x} \mapsto \mathrm{x}$ auf sich selbst an, resultiert $\mathrm{x} \mapsto \mathrm{x}$. Das ist sinnvoll. Aber der Graph von $\mathrm{x} \mapsto \mathrm{x}$ darf sich durch das Axiom of foundation nicht selbst enthalten!

(Rule-to-Graph): Nötige Zusatzannahmen

(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

Diese Behauptung ist falsch. Gegenbeispiel: $\mathrm{x} \mapsto \mathrm{x}$. Hier gibt es gleich zwei Antinomien:

1. Offensichtlich ist jeder Input für die Identitätsregel sinnvoll – auch jede Menge. Also müsste die Allmenge Teilmenge des Argumentsbereichs von $\mathrm{x} \mapsto \mathrm{x}$ sein. Aber die Allmenge existiert nicht!
2. Wendet man $\mathrm{x} \mapsto \mathrm{x}$ auf sich selbst an, resultiert $\mathrm{x} \mapsto \mathrm{x}$. Das ist sinnvoll. Aber der Graph von $\mathrm{x} \mapsto \mathrm{x}$ darf sich durch das Axiom of foundation nicht selbst enthalten!

(Rule-to-Graph): Nötige Zusatzannahmen

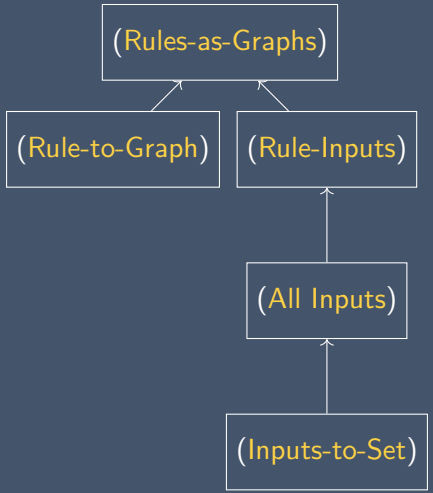
(Inputs-to-Set) Für jede Regel gilt: All ihre sinnvollen Inputs müssen (zusammen) eine Menge bilden.

Diese Behauptung ist falsch. Gegenbeispiel: $\mathrm{x} \mapsto \mathrm{x}$. Hier gibt es gleich zwei Antinomien:

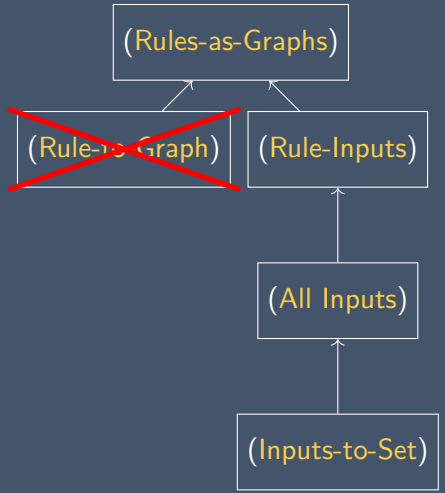
1. Offensichtlich ist jeder Input für die Identitätsregel sinnvoll – auch jede Menge. Also müsste die Allmenge Teilmenge des Argumentsbereichs von $\mathrm{x} \mapsto \mathrm{x}$ sein. Aber die Allmenge existiert nicht!
2. Wendet man $\mathrm{x} \mapsto \mathrm{x}$ auf sich selbst an, resultiert $\mathrm{x} \mapsto \mathrm{x}$. Das ist sinnvoll. Aber der Graph von $\mathrm{x} \mapsto \mathrm{x}$ darf sich durch das Axiom of foundation nicht selbst enthalten!

$\mathrm{x} \mapsto \mathrm{x}$ ist nicht die einzige Funktion, die Probleme bereitet; siehe $\mathrm{x} \mapsto \{\mathrm{x}\}$.

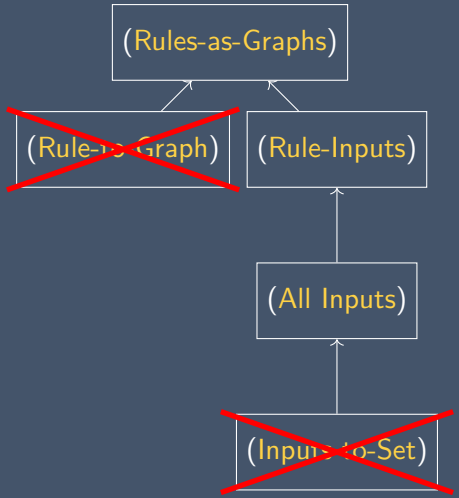
Die Probleme mit (Rule-to-Graph)



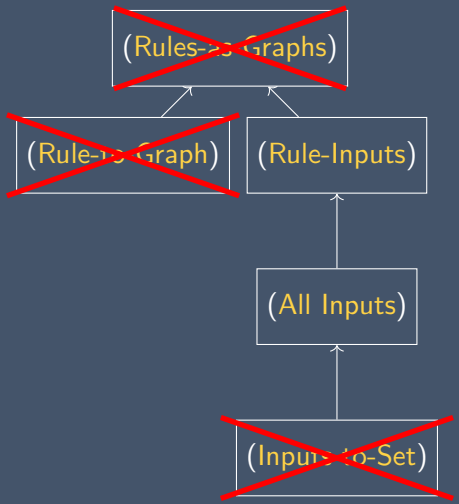
Die Probleme mit (Rule-to-Graph)



Die Probleme mit (Rule-to-Graph)



Die Probleme mit (Rule-to-Graph)



Probleme mit (Graphs-as-Rules)

Regeln als Komprehensionen?

(Graphs-as-Rules) Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

In der Mengenlehre wird die Notation für Regeln oft bei der Definition von Graphen verwendet, wie in

$(\sigma_{\text{Setrule}})$ $\sigma : \mathbb{N} \mapsto \mathbb{N}, x \mapsto x + 1$

Regeln als Komprehensionen?

(Graphs-as-Rules) Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

In der Mengenlehre wird die Notation für Regeln oft bei der Definition von Graphen verwendet, wie in

$(\sigma_{\mathrm{Setrule}})$ $\sigma : \mathbb{N} \mapsto \mathbb{N}, \mathrm{x} \mapsto \mathrm{x} + 1$

Diese Schreibweise ist erlaubt durch eingeschränkte Komprehension:

(Subset) $\forall \mathrm{x} \forall \mathrm{y} \forall \mathrm{z}(\mathrm{z} \in \mathrm{y} \leftrightarrow \mathrm{z} \in \mathrm{x} \wedge \mathrm{A}[\mathrm{z}])$

Regeln als Komprehensionen?

(Subset) $\forall x \forall y \forall z(z \in y \leftrightarrow z \in x \wedge A[z])$

Regeln lassen sich durch Variablen und das Identitätszeichen nämlich in Bedingungen umschreiben:

$(\sigma_{\text{Subset}}) \quad \sigma = \{\langle x, y\rangle \in \mathbb{N} \times \mathbb{N} \mid x + 1 = y\}$

Regeln als Komprehensionen?

(Subset) $\forall x \forall y \forall z(z \in y \leftrightarrow z \in x \wedge A[z])$

Regeln lassen sich durch Variablen und das Identitätszeichen nämlich in Bedingungen umschreiben:

$(\sigma_{\text{Subset}}) \quad \sigma = \{\langle x, y\rangle \in \mathbb{N} \times \mathbb{N} \mid x + 1 = y\}$

Wir haben also einen Graphen mithilfe einer Regel definiert! Ein Functional Egalitarian könnte das als Anlass nehmen, (Graphs-as-Rules) zu glauben.

Einzigartigkeit von Regeln

Selbst wenn (Graphs-as-Rules) stimmen würde, wäre es nicht so, dass man für jeden Graphen *genau eine* Regel findet. Zum Beispiel gibt es mehrere Regeln, mit denen sich der Graph $\{\langle 2, 4\rangle\}$ erzeugen lässt:

(1)	multipliziere den Input mit 2	$\lambda \mathrm{y}.\, 2 \times \mathrm{y}$
(2)	multipliziere den Inputs mit sich selbst	$\lambda \mathrm{y}.\, \mathrm{y} \times \mathrm{y}$
(3)	gib immer 4 als Output aus	$\lambda \mathrm{y}.\, 4$

Pseudo-Regeln für finite Graphen

Schwierig wird es, für Graphen mit willkürlich zusammengewürfelten Argument-Wert-Paaren eine Regel zu finden, wie in

(Random) $\{\langle 0, \pi \rangle, \langle 34, \mathrm{e} \rangle, \langle 135.5, 2 \rangle\}$

Pseudo-Regeln für finite Graphen

Schwierig wird es, für Graphen mit willkürlich zusammengewürfelten Argument-Wert-Paaren eine Regel zu finden, wie in

(Random) $\{\langle 0, \pi\rangle, \langle 34, \mathrm{e}\rangle, \langle 135.5, 2\rangle\}$

Ein functional egalitarian könnte antworten:

> Zeige mir irgendeinen Graphen und ich finde eine entsprechende Regeln für ihn. Ich kann dir sogar einen Algorithmus dafür angeben: Für jedes Argument-Wert-Paar $\langle \mathrm{y}, \mathrm{z}\rangle$ füge einen Konditionalsatz der Form „wenn $\mathrm{x} = \mathrm{y}$, dann $\mathrm{f}(\mathrm{x}) = \mathrm{z}$“ in die Regeldefinition ein.

Pseudo-Regeln für finite Graphen

Schwierig wird es, für Graphen mit willkürlich zusammengewürfelten Argument-Wert-Paaren eine Regel zu finden, wie in

(Random) $\{\langle 0, \pi\rangle, \langle 34, \mathrm{e}\rangle, \langle 135.5, 2\rangle\}$

Ein functional egalitarian könnte antworten:

> Zeige mir irgendeinen Graphen und ich finde eine entsprechende Regeln für ihn. Ich kann dir sogar einen Algorithmus dafür angeben: Für jedes Argument-Wert-Paar $\langle \mathrm{y}, \mathrm{z}\rangle$ füge einen Konditionalsatz der Form „wenn $\mathrm{x} = \mathrm{y}$, dann $\mathrm{f}(\mathrm{x}) = \mathrm{z}$“ in die Regeldefinition ein.

Zwar hat man hier keine *generelle* Regel, aber immerhin hat man eine Regel. Mit diesem Taschenspielertrick rettet sich der functional egalitarian.

Pseudo-Regeln und finite Graphen

(Random) $\{\langle 0, \pi\rangle, \langle 34, \mathrm{e}\rangle, \langle 135.5, 2\rangle\}$

Wendet man den Algorithmus auf (Random) an, erhält man

$$\mathrm{f}(\mathrm{x}) = \begin{cases} \pi, & \text{falls } \mathrm{x} = 0 \\ \mathrm{e}, & \text{falls } \mathrm{x} = 34 \\ 2, & \text{falls } \mathrm{x} = 135.5 \end{cases}$$

Infinite Graphen und das Auswahlaxiom

Zwar funktioniert der Algorithmus für alle endlichen Graphen. Für unendliche Graphen scheitert er aber, denn man kann eine Regel nur mit endlich vielen Bedingungen definieren.

Infinite Graphen und das Auswahlaxiom

Zwar funktioniert der Algorithmus für alle endlichen Graphen. Für unendliche Graphen scheitert er aber, denn man kann eine Regel nur mit endlich vielen Bedingungen definieren.

Die Existenz von solchen unendlichen Graphen wird aber durch das Auswahlaxiom gesichert:

(Choice) $\forall x(\emptyset \notin x \to \exists f(f : x \mapsto \bigcup x \wedge \forall y(y \in x \to f(y) \in y)))$

Infinite Graphen und das Auswahlaxiom

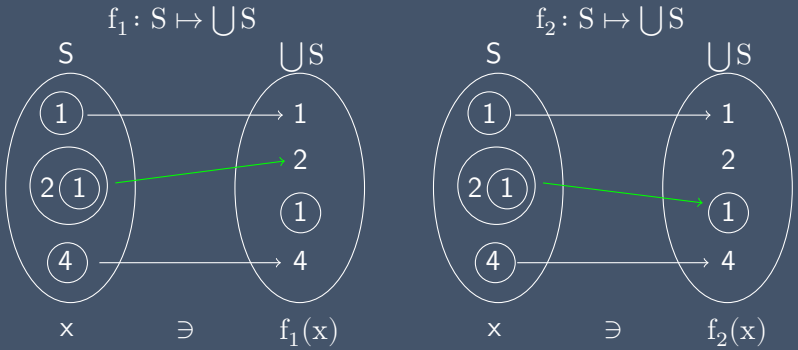
Zwar funktioniert der Algorithmus für alle endlichen Graphen. Für unendliche Graphen scheitert er aber, denn man kann eine Regel nur mit endlich vielen Bedingungen definieren.

Die Existenz von solchen unendlichen Graphen wird aber durch das Auswahlaxiom gesichert:

(Choice) $\forall x(\emptyset \notin x \to \exists f(f : x \mapsto \bigcup x \wedge \forall y(y \in x \to f(y) \in y)))$

Infinite Graphen und das Auswahlaxiom

Sei $S = \{\{1\}, \{2, \{1\}\}, \{4\}\}$. (Choice) garantiert nun die Existenz eines dieser beiden Auswahlgraphen:



Infinite Graphen und das Auswahlaxiom

Wir betrachten die Menge aller paarweise disjunkten Doubletons, deren Elemente aufeinanderfolgende natürliche Zahlen sind:

$(\mathrm{M}) \qquad \{\{0,1\},\{2,3\},\{4,5\},...\}$

Da $\emptyset \notin \mathrm{M}$, gilt (Choice) für M. Es garantiert die Existenz von einem der $2^{|\mathrm{M}|} = |\mathbb{N}|$ möglichen Choice-Graphen. Wir wissen aber nicht, welcher es ist. Wir wissen nur, dass seine Argumentmenge M ist und seine Wertmenge die der natürlichen Zahlen.

Infinite Graphen und das Auswahlaxiom

Nennen wir den Choice-Graphen für M, der durch (Choice) garantiert ist, c. c ist selbst eine Menge von Doubletons, denn geordnete Paaren sind nach Kuratowski, 1921 Doubletons:

(Pair) $\langle \mathrm{x}, \mathrm{y} \rangle = \{\{\mathrm{x}, \mathrm{y}\}, \{\mathrm{x}\}\}$

Infinite Graphen und das Auswahlaxiom

Nennen wir den Choice-Graphen für M, der durch (Choice) garantiert ist, c. c ist selbst eine Menge von Doubletons, denn geordnete Paaren sind nach Kuratowski, 1921 Doubletons:

(Pair) $\langle \mathrm{x}, \mathrm{y} \rangle = \{\{\mathrm{x}, \mathrm{y}\}, \{\mathrm{x}\}\}$

Offensichtlich ist $\emptyset \notin \mathrm{c}$. Also existiert auch ein Choice-Graph für den Choice-Graphen von M.

Infinite Graphen und das Auswahlaxiom

Nennen wir den Choice-Graphen für M, der durch (Choice) garantiert ist, c. c ist selbst eine Menge von Doubletons, denn geordnete Paaren sind nach Kuratowski, 1921 Doubletons:

(Pair) $\quad \langle \mathrm{x}, \mathrm{y} \rangle = \{\{\mathrm{x}, \mathrm{y}\}, \{\mathrm{x}\}\}$

Offensichtlich ist $\emptyset \notin \mathrm{c}$. Also existiert auch ein Choice-Graph für den Choice-Graphen von M.

Damit ist die Existenz eines infiniten Graphen gesichert, von dem wir weder seinen Argumentbereich noch ein einziges Paar kennen. Eine Regel lässt sich für diesen Graphen aber nicht aufstellen! Das widerspricht (Graphs-as-Rules).

Intuitionismus zur Rettung?

Da (Choice) Probleme für den funcitonal egalitarian erzeugt, könnte er oder sie es einfach ablehnen und eine intuitionistische Ansicht vertreten. Aber selbst Intutionist:innen haben eine Spielform von (Choice)

(Countable Choice) $\forall x(\emptyset \notin x \wedge |x| \leq |\mathbb{N}| \rightarrow \exists f(f : x \mapsto \bigcup x \wedge \forall y(y \in x \rightarrow f(y) \in y)))$

Intuitionismus zur Rettung?

Da (Choice) Probleme für den funcitonal egalitarian erzeugt, könnte er oder sie es einfach ablehnen und eine intuitionistische Ansicht vertreten. Aber selbst Intutionist:innen haben eine Spielform von (Choice)

(Countable Choice) $\forall x(\emptyset \notin x \wedge |x| \leq |\mathbb{N}| \rightarrow \exists f(f : x \mapsto \bigcup x \wedge \forall y(y \in x \rightarrow f(y) \in y)))$

(Countable Choice) reicht aber aus, um das obige Problem zu erzeugen. Also ist auch (der Standard-)Intuitionismus kein Ausweg für den functional egalitarian!

Diskussion und Fazit

Fazit

(Rules-as-Graphs) Für jede Regel lässt sich ein ihr entsprechender Graph konstruieren.

(Graphs-as-Rules) Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

Fazit

(Rules-as-Graphs)	Für jede Regel lässt sich ein ihr entsprechender Graph konstruieren.
(Graphs-as-Rules)	Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

Wir haben gerade gezeigt, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. In beiden Fällen sind die mengentheoretischen Axiome ein Problem: Für (Rules-as-Graphs) Das Fundierungsaxiom und das eingeschränkte Komprehensionsschema, für (Graphs-as-Rules) das Auswahlaxiom.

Fazit

(Rules-as-Graphs)	Für jede Regel lässt sich ein ihr entsprechender Graph konstruieren.
(Graphs-as-Rules)	Für jeden Graphen lässt sich eine ihm entsprechende Regel konstruieren.

Wir haben gerade gezeigt, dass sowohl (Rules-as-Graphs) als auch (Graphs-as-Rules) falsch sind. In beiden Fällen sind die mengentheoretischen Axiome ein Problem: Für (Rules-as-Graphs) Das Fundierungsaxiom und das eingeschränkte Komprehensionsschema, für (Graphs-as-Rules) das Auswahlaxiom.

Fazit

Es lässt sich also festhalten:

1. Es gibt Regeln, für die kein entsprechender Graph existiert.

Fazit

Es lässt sich also festhalten:

1. Es gibt Regeln, für die kein entsprechender Graph existiert.
2. Es gibt Graphen, für die keine entsprechende Regel existiert.

Fazit

Es lässt sich also festhalten:

1. Es gibt Regeln, für die kein entsprechender Graph existiert.
2. Es gibt Graphen, für die keine entsprechende Regel existiert.

Fazit

Es lässt sich also festhalten:

1. Es gibt Regeln, für die kein entsprechender Graph existiert.
2. Es gibt Graphen, für die keine entsprechende Regel existiert.

Je nachdem, ob man (Functions-as-Graphs) oder (Functions-as-Rules) wählt, bekommt man also andere Funktionen. Damit ist Functional Egalitarianism keine haltbare Position – und die Entscheidung, ob (Functions-as-Graphs) oder (Functions-as-Rules) gilt, nicht nur ontologisch, sondern auch formal relevant!

Diskussion

1. Sind meine Argumente jeweils schlüssig?

Diskussion

1. Sind meine Argumente jeweils schlüssig?
2. Gibt es weitere Argumente gegen den Functional Egalitarian?

Diskussion

1. Sind meine Argumente jeweils schlüssig?
2. Gibt es weitere Argumente gegen den Functional Egalitarian?
3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?

Diskussion

1. Sind meine Argumente jeweils schlüssig?
2. Gibt es weitere Argumente gegen den Functional Egalitarian?
3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?
4. An welchen Stellen drücke ich mich unklar aus?

Diskussion

1. Sind meine Argumente jeweils schlüssig?
2. Gibt es weitere Argumente gegen den Functional Egalitarian?
3. Nenne ich Behauptungen, zu denen ein anderer Name passender wäre?
4. An welchen Stellen drücke ich mich unklar aus?
5. Gibt es nicht vielleicht doch ein Schlupfloch für den Functional Egalitarian?

Literatur

Kuratowski, C. (1921). Sur la notion de l'ordre dans la Théorie des Ensembles. *Fundamenta Mathematicae*, *2*(1), 161–171 (siehe S. 92–94).