Dr. Christoph Lübbert
Viktoriastraße 36
D-64293 Darmstadt

Tel: 06151 422298, T-Mob: 0171 2045811,
christoph-luebbert@t-online.de ,
www.cl-diesunddas.de

Darmstadt, 26.02.2025

Abschlussbericht im RW-Seminar am 27.02.2025 zum Thema

„Mathematisierungsversuch für UNFOLD“,

der seit Herbst 2022 bestehenden Diskussionsgruppe
***Hermann Bense* (HB), *Thomas Zeh* (TZ), *Christoph Lübbert* (CL)**

Akronym „**UNFOLD**“:
„**UN**ifying **F**ramework for **O**ntology and **L**inguistics **D**evelopment“
https://www.taoke.de/
ist die seit vielen Jahren laufende *Workbench* von Hermann Bense
mit hunderten von WebSites.

0 Anlass & vorbereitende Diskussionen

Hermann Bense (HB) bat uns (TZ, CL) im Sommer 2022, sein Werk UNFOLD zu „mathematisieren“. Er wollte seinen **Ontology-Modellierungen** eine mathematische Grundlage geben, die der Denkweise der **FBA** (Formale Begriffsanalyse) möglichst nahekommt.

Wir (TZ, CL) stimmten freudig zu, ohne noch eine Ahnung davon zu haben, welche **sprachlichen** und **notationsmäßigen Hürden** da auf uns zukommen würden.

Wir merkten bald, dass die Diskussion „*ganz von vorne*“ beginnen müsse. Erst mit der Zeit kamen wir dann *schrittweise* auf **HBs spezielle Wünsche** für UNFOLD. Folgende Vorbereitungen schienen mir für den „Mathematisierungsversuch“ zunächst einmal erforderlich:

*1 **PL1(=)** – **Prädikatenlogik 1.Stufe mit „Gleichheit „=“:** Syntax & Semantik.
*2 **Mengenbegriff, Mengenschreibweise:** In UNFOLD stets nur für **endliche** Mengen.
*3 Binäre Relationen, Funktionen, Ordnungs- und Verbandsbegriff.
*4 **Konzept FBA („Formale Begriffsanalyse“):** „Vollständiger Verband“, „Formaler Kontext“, „Formaler Begriff“, „Begriffsverband“, „Liniendiagramm“, „Dualität“ zwischen der Menge der „Begriffs-**Umfänge**“ und der Menge der „Begriffs-**Inhalte**“ etc. …

Dabei war ***von vorne herein*** zu berücksichtigen, dass stets von den UNFOLD-**Beispielen** und von der in UNFOLD herrschenden „**Methodology**“ auszugehen sei.

– Einfach mathematische Definitionen anzugeben, wäre **viel zu weit entfernt** gewesen von informatischer Denk-, Schreib- und Sprechweise in UNFOLD!

1 Vorbereitendes Beispiel – Formaler Kontext / Begriffsverband

Hier ein vorbereitendes (formales) Beispiel, das sowohl die in UNFOLD gebräuchliche Darstellungsweise für das Modell „**Onto“**, als auch das in FBA übliche „**Liniendiagramm**“ für einen (endlichen u. damit vollständigen) **Verband** andeutet.

Man geht aus von einem **formalen Kontext**

K = (G, M, I) mit **G =**Menge der **Gegenstände, M =**Menge der **Merkmale,**
I = die **Inzidenzrelation** (wenn $x \in G$, $a \in M$, $(x,a) \in I$, dann Kreuzchen in Zelle (x, a); andernfalls kein Kreuzchen)

Für jeden **formalen Begriff B = (P, A)** mit **Umfang** $P \subseteq G$, **Inhalt** $A \subseteq M$ gelten die „Ableitungsregeln“

$$\mathbf{P^{\uparrow} = A} \text{ und } \mathbf{A^{\downarrow} = P.}$$

Dabei entsprechen im „**Onto**“-Modell die ***Umfänge*** P den **Klassen**, die ***Inhalte*** A den zugehörigen **Attributmengen** (DP-Mengen) und das Liniendiagramm der **Klassenhierarchie**.

Beispiel: Fomaler Kontext / Begriffsverband

CL 23.2.2025

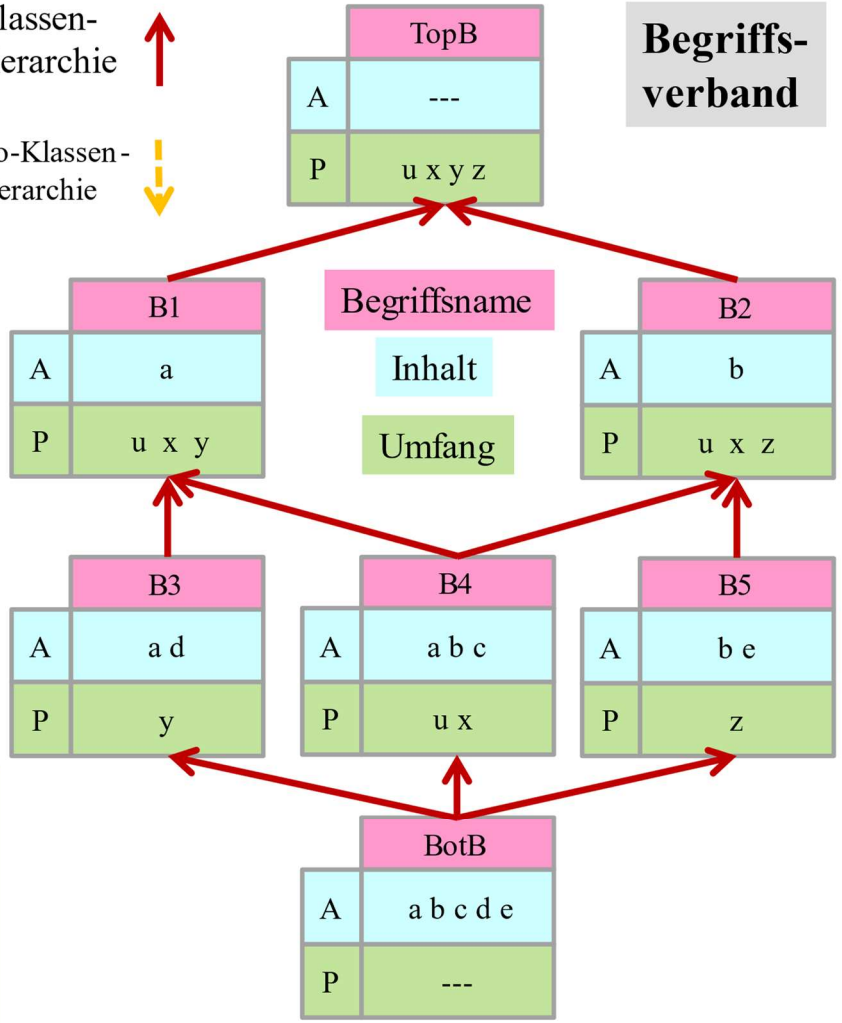
Formaler Kontext

Gegenstände (*Particulars*)	Merkmale (*Attributes*) a	b	c	d	e
u	x	x	x		
x	x	x	x		
y	x			x	
z		x			x

Klassen-hierarchie

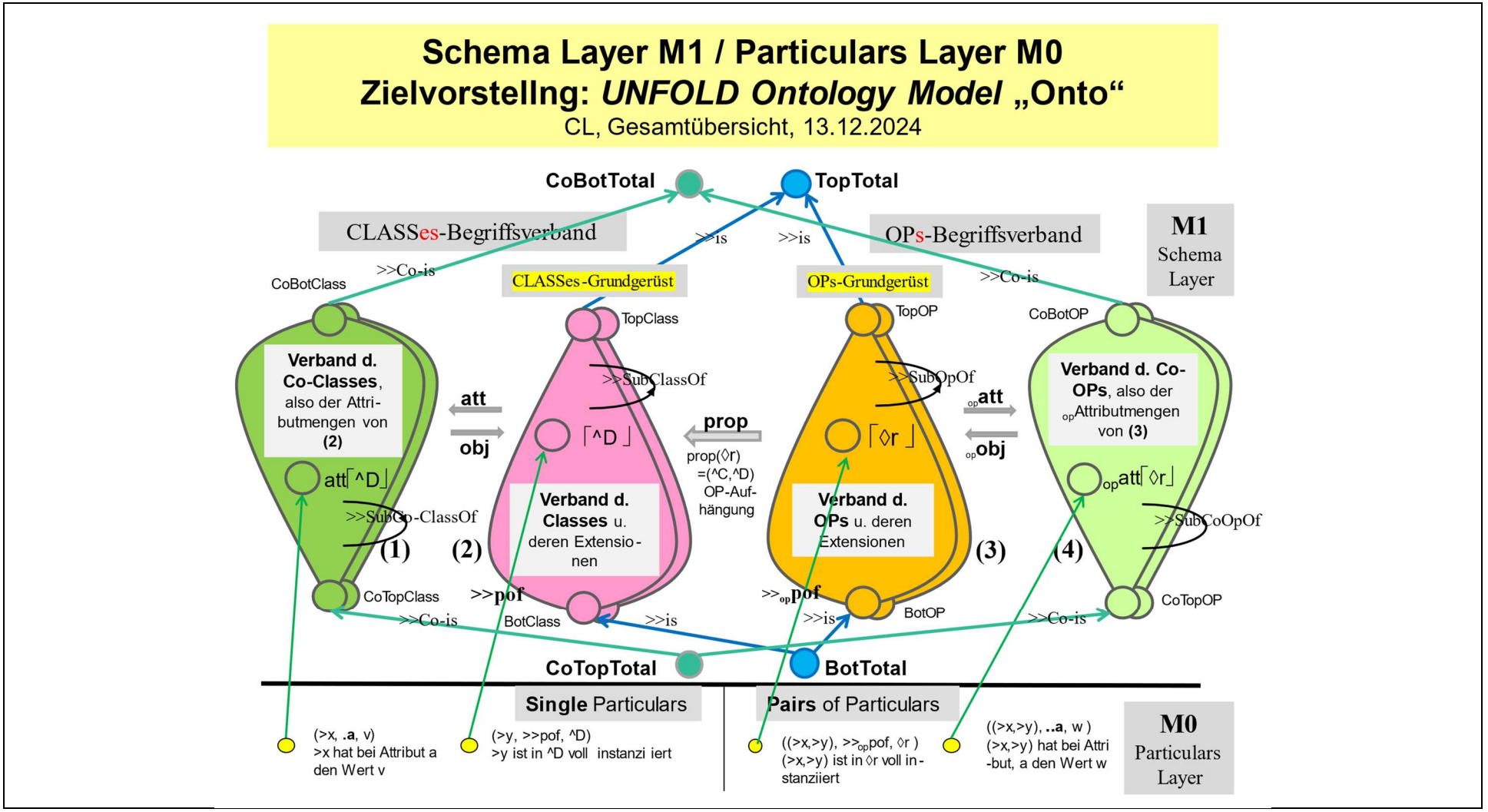
Co-Klassen-hierarchie

Begriffs-verband



FBA	UNFOLD
Gegenstände	*Particulars*
Umfänge	*Class Extensions*
Merkmale	*Attributes (Data Properties)*
Inhalte	*Class Intensions*

2 Meine Zielvorstellung für UNFOLD



Aus der eben gezeigten Ziel-Übersicht bespreche ich ausführlicher hier nur den **linken Teil (1) dunkelgrün** u. **(2) rot** für **Klassen & Particulars**.

Der rechte Teil (3) **orange** u. (4) **hellgrün** für „**Object Properties**“ & **Particular-Paare** geht ganz analog.

3 „Top-down-Ansatz“ für Klassen & Particulars

Vom „*Abstakteren*“ im Schema-Layer **M1** zum „*Konkreteren*“ im Particules-Layer **M0**.

($\mathscr{C}$, $\leq_{CH}$, OBJ, ext, ATTR, att, Attribut*vererbung,* Attribut*deklaration*)

- „**Grundgerüst**“: ($\mathscr{C}$, $\leq_{CH}$) ist endliche, geordnete Menge, deren Elemente „**Klassen**“ heißen, und deren Ordnung $\leq_{CH}$ die „**Klassenhierarchie**“ heißt. In UNFOLD wird die Klassenhierarchie mit **>>SubClassOf** notiert, also:
  ^C $\leq_{CH}$ ^D $\Leftrightarrow$ **(^C, >>SubClassOf, ^D)** – („Tripelschreib-weise“ in UNFOLD).
- **OBJ** ist endliche die Menge der untersuchten sog. „**Particulars**“ („Einzeldinge“). Klassen und Particulars sind **streng zu trennen:** *Kein Particular ist eine Klasse, keine Klasse ist ein Particular,* $\mathscr{C} \cap$ **OBJ =** $\varnothing$**.**
  Klassen bevölkern den sog. **Schema Layer M1**, Particulars den sog. **Particulars Layer M0**.
- **Extension: ext:** $\mathscr{C} \rightarrow$ Pot(**OBJ**) ist eine **Funktion**, die jeder Klasse ^C eine Particular-Menge **ext(^C)**$\subseteq$OBJ eindeutig zuordnet, **derart, dass gilt**
  $\forall$**^C,^D**$\in\mathscr{C}$**: ^C** $\leq_{CH}$ **^D** $\Leftrightarrow$ **ext(^C)** $\subseteq$ **ext(^D) ;** dann ist auch

$\forall$**^C**$\in\mathscr{C}$**:** $\mathscr{C} \cap$ **ext(^C) =** $\varnothing$**.** ext(^C) nennen wir die **Extension** der Klasse ^C in den Particulars-Layer **M0** hinein.

- **ATTR** ist die Menge aller „**Attribute**“ (in UNFOLD „**Data Properties**“ (DP) genannt, in FBA „**Merkmale**“ genannt). „Attribute“ sind dazu gut, die „Objekte“ zu *charakterisieren*.
- **att: CLAP** $\rightarrow$ Pot(**ATTR**) mit der Abkürzung **CLAP:=** $\mathscr{C} \cup^{\bullet}$ **OBJ. att** ist eine Funktion, die jedem Objekt $\chi\in$CLAP – also jeder Klasse ^X$\in\mathscr{C}$ bzw. jedem Particular >x$\in$OBJ – eindeutig ihre Attributmenge (DP-Menge) att(^C) bzw. att(>x) zuordnet.
  Beachte: In UNFOLD werden **sowohl** den **Klassen als auch** den **Particulars** gewisse Attributmengen zugeschrieben. Daher haben wir $\mathscr{C} \cap$ **OBJ =** $\varnothing$ gesetzt und daher haben wir die **Extension** der Klassen eingeführt.
- **Attributvererbung** (DP-Vererbung)**:** Das Wort „Vererbung“ (*inheritance*) stammt aus der **OOP** (***o**bject-**o**riented **p**rogramming*). Darunter verstehen wir hier einfach Folgendes:
  $\forall$^C,^D$\in\mathscr{C}$: ^C $\leq_{CH}$ ^D $\Leftrightarrow$ (att(^D)$\subseteq$att(^C) $\wedge$ **∩**att[ext(^D)] $\subseteq$ **∩**att[ext(^C)]).
  Erläuterung: ∩att[ext(^D)] ist der Durchschnitt der Attributmengen att(>x) aller Parculars >x$\in$ext(^D), d.h. es sind diejenigen Attribute, die allen Particulars >x$\in$ext(^D) ***gemeinsam*** sind.
  Sprechweise: Ein Attribut aus einer Klasse ^D oder deren **Extension** wird „von oben nach unten“ an alle *Unterklassen* ^C von ^D (bzw. deren Extensionen) „**vererbt**“. Das entspricht genau der Kurz-Lesweise eines FBA **Liniendiagramms**, wo an jedem „Knoten“ (= formalen Begriff) oberhalb des Begriffsknotens nur noch die „***neuen*** Attribute“ angegeben sind, aber nicht mehr die von oben vererbten).
  - **Attribut-Deklaration:** Ist ^D_a die (gemäß der Klassenhierarchie $\leq_{CH}$) **größte** Klasse, in der (oder deren **Extension** ext(D_a)) ein Attribut **a**$\in$ATTR vorkommt, so wird **a** (wie wir sagen) ***„in der Klasse ^D_a deklariert“***. D.h.: Im Klassenkästchen von ^D_a wird der Attributname „**a**“

und ihr **Wertebereich** W_a angegeben (ein W_a ist z.B. vom Typ `:String, :Integer, :Float, :Boolean,` u.a.m). – ^D_a heißt die „**Deklarationsklasse**“ des Attributs **a**. Ab ^D_a kann das Attribut **a** dann an alle Unterklassen von ^D_a „**vererbt**“ werden.

3.1 Charakterisierung der Klassen und ihrer Extensionen

Kurz gesagt: ***Je größer eine Klasse*** *in der Klassenhierarchie* $\leq_{CH}$, ***desto weniger Attribute hat sowohl die Klasse als auch ihre Extension. Je kleiner eine Klasse*** *in der Klassenhierarchie* $\leq_{CH}$, ***desto mehr Attribute hat sowohl die Klasse als auch ihre Extension.***
(Diese Einsicht ergibt sich schlicht aus der OOP-Attributvererbung. Dazu braucht man nicht einmal den vollen formalen Apparat der FBA.)

Daraus ergibt sich:

Zwei Klassen ^C, ^D sind *gleich* genau dann, wenn ihre Attributmengen gleich *und* die Attributmengen ihrer Extensionen gleich sind.

Das heißt: Jede Klasse $^C \in \mathscr{C}$ wird durch ihre Attributmenge att(^C), bzw. ihre Extension wird durch die Attributmenge **⋂**att**[**ext(^C)**]** **eindeutig charakterisiert.**

Anm.: Als ich das vor etwa 15 Jahren einmal im „Ontology-Kreis“ der Hochschule Darmstadt (***hda***) vortrug, **protestierten** die Kollegen der *hda* und meinten: ein solcher Klassenbegriff sei *viel* ***zu einschränkend*** für sie. – In der Nachdiskussion ergab sich jedoch, dass die werten Herren schlicht einige Datenparameter ihres Klassenbegriffs zu **übersehen** pflegen. (Das ist leider jene „schnodderige“ Ausdrucksweise, in der **Tonnen** von Ontology-Artikeln hingeschrieben sind.)

3.2 Die Attribut-Typen in UNFOLD

Jedes Attribut (Merkmal, DP) **a**∈ATTR ist eine **Funktion**

a: dom(a) → W_a des **Definitionsbereichs** dom(**a**) in einen Wertebereich **W_a** (`:String, :Integer, :Float, :Boolean, :Gender,` u.a.m). In UNFOLD unterscheidet man **drei** untereinander **fremde Attributtypen** (DP-Typen):

ATTR := PDP ∪• MDP ∪• UDP.

Sei dom(**a**) der Definitionsbereich des Attributs (DPs) **a**, und ^D_a sei die Deklarationsklasse für **a**.

* **PDP** (**Particular Data Property**): Ein DP •**a**∈PDP hat den Definitionsbereich

dom(•**a**) = ext(^D_a). **nur** Particulars tragen PDPs, Klassen selbst tragen **keine** PDPs.

* **MDP** (**Meta Data Property**): Ein DP •^^**a**∈MDP hat den Definitionsbereich

dom(•^^**a**) = ∴**^D_a** ∪• **ext(^D_a)** (∴**^D_a** = Hauptideal der Klasse ^D_a)

sowohl Klassen **als auch** Particulars können MDPs tragen.

* **UDP** (**Universal Data Property**): Ein DP •^**a**∈UDP hat den Definitionsbereich

dom(•^**a**) = ∴**^D_a** **nur Klassen** können UDPs tragen, Particulars nicht.

Anm: Ihr habt wohl schon bemerkt, dass gewisse **Prefix-Zeichen** vor Objektnamen, Attributnamen und Relatoren gesetzt sind. Sie dienen in UNFOLD der **schnelleren Lesbarkeit:**

^ vor Klassennamen, > vor Particular-Namen, • vor PDPs, •^^ vor MDPs, •^ vor UDPs.

Analog:

◊ vor Object Properties, (>x,>y) bei Particular-Paaren, •• vor $_{op}$PDPs, ••^^ vor $_{op}$MDPs, •^ vor $_{op}$UDPs.

Allgemein: >> vor einem „Relator“ (d.i. der Name für ein Prädikatsymbol), **:** vor Wertebereichen.

3.3 Instanziierung

Gebrauch und Bedeutung der Worte „**Instanziierung**“, „**Instanz**“ ist bei den mit „Ontology“ befassten Informatikern ***uneinheitlich***. Wir haben uns in der UNFOLD-Diskussion auf folgenden Gebrauch geeinigt:

Jedes Attribut **a**∈ATTR liefert für jedes Objekt $\chi \in dom(\mathbf{a}) \subseteq$ **CLAP** einen bestimmten **Wert** $\mathbf{a}(\chi) \in W_a$. Den Vorgang der Wertzuweisung $\chi \rightarrow \mathbf{a}(\chi)$ nennen wir „**Instanziierung**“. Das Ergebnis $\mathbf{a}(\chi)$ nennen wir die „**Instanz**“ des Objekts χ beim Attribut **a**.

Ein bestimmtes Particular >x∈OBJ heißt „in der Klasse ^C **voll instanziiert**“, wenn
att(>x) = ∩att[(ext(^C)] ist. ^C ist für gegebenes >x **eindeutig bestimmt**: ^C ist nämlich die (in der Klassenhierarchie $\leq_{CH}$) ***kleinste*** Klasse, mit x∈ext(^C). In UNFOLD hat man dafür die Aussage-Tripelform

(>x, >>**pof**, ^C) :⇔ *„particular >x is fully instantiated in class ^C“.*

Anm.: Zu den **eckigen** Klammen [ ] in att**[**(ext(^C)**]:** Sei z.B. ext(^C)= {$>x_1$, $>x_2$, $>x_3$}; dann ist att**[**(ext(^C)**] := att**{$>x_1$, $>x_2$, $>x_3$} := {att($>x_1$), att($>x_2$) att($>x_3$)} eine **Menge** von Attribut**mengen.**
Erst der **Durchschnitt ∩**att**[**(ext(^C)**]** := att($>x_1$)∩att($>x_2$)∩att($>x_3$) ist wieder *eine* Attribut*menge*!
Dies ist Mathematikern geläufig, aber für UNFOLD musste es *besonders betont* werden.

Bottom-up-Ansatz für Klassen & Particulars

Im sog. „Bottom-Ansatz“ orientieren wir uns näher an der Notations- u. Denkweise der **FBA** (Formale Begriffsanalyse).

Anm.: Zur Funktion

att: CLAP → Pot(ATTR),

die jedem Objekt $\chi \in$ CLAP (also einer Klasse od. einem Paritcular) einen **Inhalt**

att(χ) := $\{a \in ATTR \mid (\chi, a) \in iRel_{CLAP}\}$

eindeutig zuordnet, haben wir ***dual*** dazu die Funktion

obj: ATTR → Pot(CLAP),

die jedem Attribut **a** $\in$ ATTR einen **Umfang** (u. somit eine Klasse)

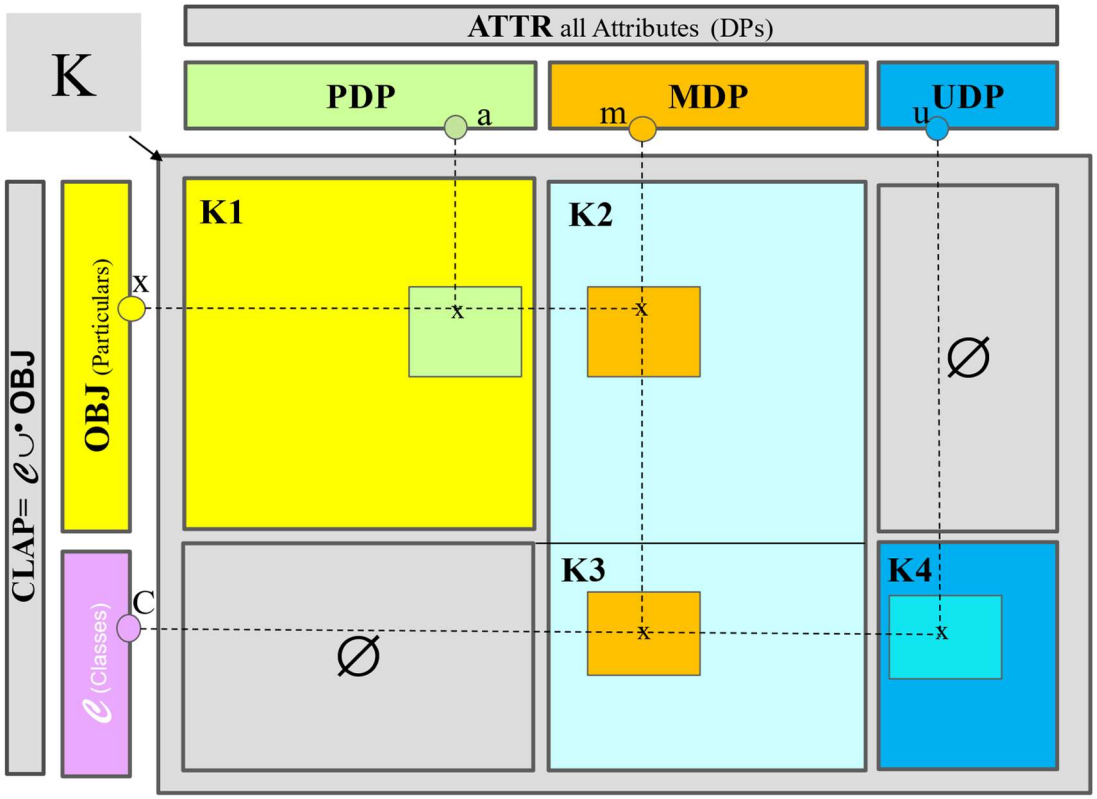
obj(a) := $\{\chi \in CLAP \mid (\chi, a) \in iRel_{CLAP}\}$

eindeutig zuordnet.

Auch dies musste für UNFOLD *besonders betont* werden.

Formaler Gesamt-Kontext K:= K_{CLAP} für Klassen & Particulars und ein paar (nicht-leere) Teilkontexte K1, K2, K3, K4

CL23.02.2025

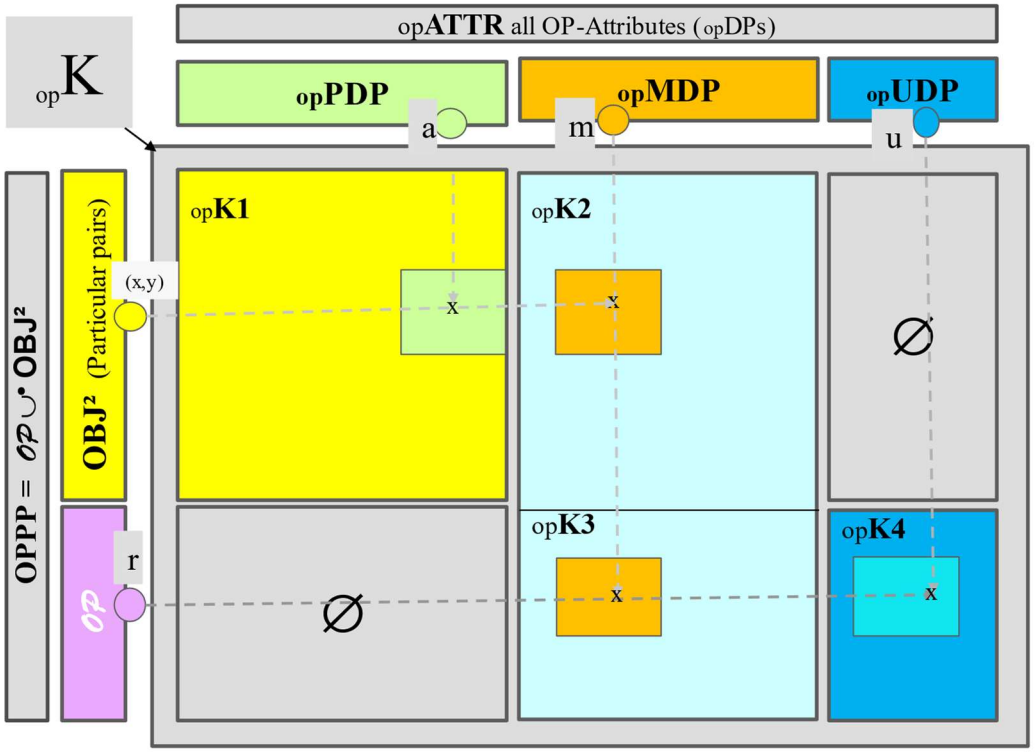


5 Bottom-up-Ansatz für Object Properties (OPs) & Particular-Paare

Es geht **alles analog** wie im Bottom-up-Ansatz für Klassen & Particulars.

Formaler Gesamtkontext $_{op}K = K_{OPPP}$ für Object Properties & Particular Pairs und ein paar (nicht-leere) **Teilkontexte $_{op}K1$, $_{op}K2$, $_{op}K3$, $_{op}K4$.**

CL23.02.2025



6 Aufhängung von Object Properties (OP) im Klassen-Grundgerüst

Die **Grundgerüste** ($\mathcal{C}$**,** $\leq_{CH}$) für Kassen u. ($\mathcal{OP}$**,** $\leq_{OP}$) für Object Properties sind **unabhängig von einander**. Das Einzige, was beide Grundgerüste verbindet, ist die Menge **OBJ** der im „Onto"-Modell betrachteten **Particulars**.

* In der **Extension einer Klasse** treten **Einzel-Particulars** $>x \in OBJ$ auf,
* in der **Extension eines OP** treten **Particular-Paare** $(>x, >y) \in OBJ^2 = OBJ \times OBJ$ auf.

Will man ein OP $\Diamond$**r**$\in \mathcal{OP}$ im ***selben*** Diagramm zusammen mit dem Grundgerüst ($\mathcal{C}$**,** $\leq_{CH}$) der Klassen darstellen, so muss man wissen, **wo $\Diamond$r einzuzeichnen sei**.

Die Funktionsgleichung **prop($\Diamond$r) = (^C, ^D)** besagt: Das *OP* $\Diamond$**r** heißt *„im Klassen-Paar (^C, ^D) aufgehängt"* vermöge der Definition:

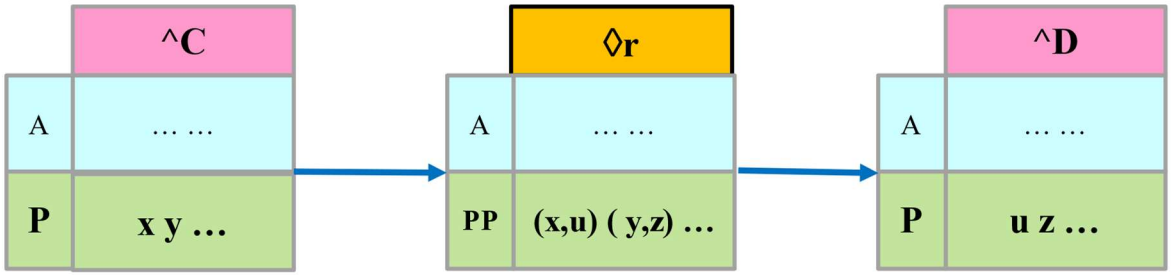
^C ist die (gemäß $\leq_{CH}$) **kleinste** Klasse mit **dom(ext($\Diamond$r)) $\subseteq$ ext(^C)**, und
^D ist die (gemäß $\leq_{CH}$) **kleinste** Klasse mit **range(ext($\Diamond$r)) $\subseteq$ ext(^D).**

Anm: Wo die beiden Aufhängungs-Klassen ^C, ^D in der Klassenhierarchie $\leq_{CH}$ liegen, ist ***unerheblich***. Es kann auch ^C = ^D sein. Es kann auch sein, dass zwei (oder mehr) verschiedene OPs $\Diamond$**r**, $\Diamond$**s** im **selben** Klassenpaar (^C, ^D) aufgehängt sind. Es bedarf **keiner „Verbote"**, wie sie (anfänglich, aus Unkenntnis d. math. Struktur) in UNFOLD gefordert wurden!

Aufhängung eines Object Property ◊r zwischen zwei Klassen ^C, ^D

CL 23.02.25

OP-Aufhängung hat nichts mit Attributen zu tun.
Daher ist hier bei den Inhalten „A“ auch nichts eingetragen.

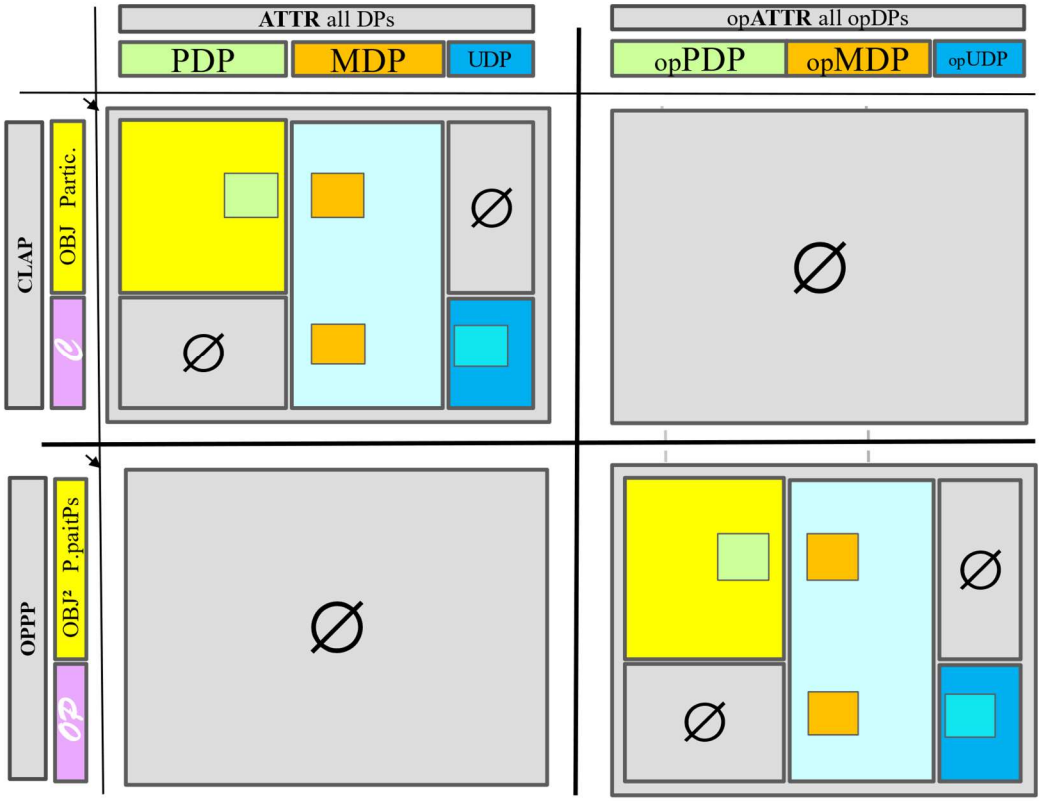


prop(◊r) = (^C, ^D) bedeutet :
^C bzw. ^D sind die jeweils kleinsten Klassen mit
dom(ext(◊r)) ⊆ ext(^C) und range(ext(^D)) ⊆ ext(^D)

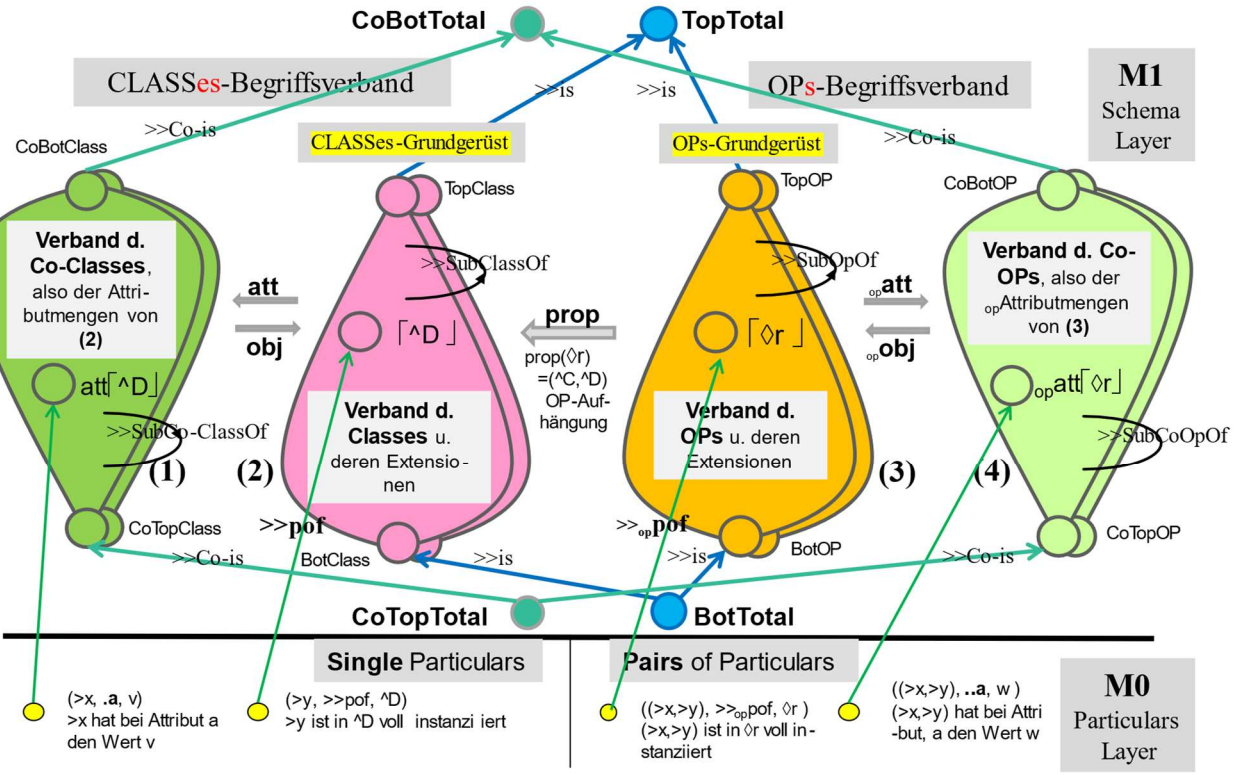
7 Schlussbilder – Total-Kontext & Total-Begriffsvervand für das UNFOLD „Onto“ Model

Formaler Total-Kontext für UNFOLD

disjunkte Vereinigung $K_{total} = K_{CLAP} \dot{\cup} K_{OPPP}$ CL 04.12.2024



Dieser Total-Kontext entspricht der anfänglich gezeigten Zielvorstellung für den Total-Begriffsverband.



Ich danke für Eure Geduld – CL