Developing an Ontological Sandbox:

Investigating Multi-Level Modelling's Possible Metaphysical Structures

Chris Partridge[1,2] [0000-0003-2631-1627], Sergio de Cesare[1] [0000-0002-2559-0567], Andrew Mitchell[2] [0000-0001-9131-722X], Frederik Gailly[3] [0000-0003-0481-9745], Mesbah Khan[4] [0000-0002-1327-6263]

[1] University of Westminster, [2] BORO Solutions Ltd., [3] Faculty of Economics and Business Administration, Ghent University, [4] Tullow Oil plc.

Abstract. One of the central concerns of the multi-level modelling (MLM) community is the hierarchy of classifications that appear in conceptual models; what these are, how they are linked and how they should be organised into levels and modelled [1]. Though there has been significant work done in this area, we believe that it could be enhanced by introducing a systematic way to investigate the ontological nature and requirements that underlie the frameworks and tools proposed by the community to support MLM (such as Orthogonal Classification Architecture and Melanee). In this paper, we introduce a key component for the investigation and understanding of the ontological requirements, an ontological sandbox. This is a conceptual framework for investigating and comparing multiple variations of possible ontologies – without having to commit to any of them – isolated from a full commitment to any foundational ontology. We discuss the sandbox framework as well as walking through an example of how it can be used to investigate a simple ontology. The example, despite its simplicity, illustrates how the constructional approach can help to expose and explain the metaphysical structures used in ontologies, and so reveal the underlying nature of MLM levelling.

Keywords: Ontological Sandbox · Constructional Ontology · Ontological Space · Ontogenesis · Multi-Level Modelling · Generalisation · Classification

1 Introduction

One of the central concerns of the multi-level modelling (MLM) community is the hierarchy of classifications that appear in conceptual models; what these are, how they are linked and how they should be organised into levels and modelled [1]. It has been recognized that these levels are sometimes ontological [2] and, where they are, that this introduces constraints, such as anti-cyclicity. There have also been attempts to characterise classification and how it differs from generalisation [3–5] that include consideration of their different ontological natures. Though there has been significant work done in this area, we believe that it could be enhanced by introducing a systematic way to

investigate the ontological nature and requirements that underlie the levels and so inform the frameworks and tools proposed by the community to support MLM (such as Orthogonal Classification Architecture and Melanee).

In the long term, we aim to provide support for the investigation and understanding of the ontological requirements and so guide the design of ontologies including those used in MLM frameworks and tools. In this paper, we introduce a key component of this, an ontological sandbox. This is a conceptual framework for investigating and comparing multiple variations of possible ontologies – without having to commit to any of them – isolated from a full commitment to any foundational ontology. The sandbox helps to expose and explain the metaphysical structures of the candidate ontologies, the differences between them as well as suggesting potential alternatives. This makes it useful for assessing different architectural choices.

Here we aim to provide a sketch of the sandbox based upon a constructional approach outlined in [6, 7]. We discuss the sandbox framework as well as walking through an example of how it can be used to investigate a simple ontology. The example, despite its simplicity, illustrates how the constructional approach can help to expose and explain the metaphysical structures found in ontologies, and so reveal the underlying nature of MLM levelling. It illustrates how metaphysical choices [8] guide the construction of the ontology and how understanding the way metaphysical structures can vary helps to guide the investigation of possible ontologies.

We look at how to use these metaphysical structures to derive, and so help to explain the kinds of levelled structures frequently found in conceptual modelling; including taxonomic structures such as the Linnaean hierarchy and component breakdowns common in engineering. This reveals a common underlying foundation for the levelled structures that suggests there may be a wider application for MLM techniques.

We look at the MLM structures themselves. We use the example as a foundation for characterising the family of formal structures associated with MLM classification and generalisation. We show how understanding the way they are derived from fundamental structures helps to explain what they are and how different derivations give rise to the variety of members of the family.

2 The Sandbox's Underlying Framework

The sandbox adopts the algebraic constructional ontological framework outlined by Fine in [6] and further developed, with a focus on wholes and parts, in [7]. Fine sees the advantage of his framework is that it naturally reveals the underlying metaphysical structure of reality. As an example, he comments on levels: "there is an intuitive distinction between wholes which are like sets in being hierarchically organised and those which are like sums in being 'flat', or without an internal division into levels. The distinction, under the operational approach, can be seen to turn on whether repeated applications of the operation are capable of yielding something new." [7, p. 566]. (One can see the same flat/hierarchy distinction made in the literature between generalisation and classification – a topic we return to below.) In the same paper, he talks about its power and beauty; its ability to provide a single and elegant account of a variety of structures.

One cannot see this in the logical characterisation to these hierarchies, such as [9]. This makes it a better tool for the task of investigating the metaphysical structure of possible ontologies as well as the ontological content of MLM's levels.

Here we outline, with some minor clarifications, the relevant portion of Fine's framework. This is not intended to be a detailed exposition; this can be found in Fine's papers. We broadly follow Fine's notation, with some amendments to make this short exposition clearer. We divide the outline into two sections; the first dealing with the general ontological framework and the second with the general composition framework.

2.1 Finean General Ontological Framework

Fine's general framework has two theories; a core theory about ontologies and an extended theory for ontological spaces containing ontologies.

Fine's core theory deals with constructional ontologies. (From now on we will feel free to drop the qualification 'constructional' from ontology – as all ontologies discussed here will be constructional in the Finean sense.) In these ontologies, objects are accepted into the ontology on the grounds that they are constructed from elements already in the ontology; where a constructor is applied to the constructees to produce constructs. An ontology can also contain given or basic elements that are just accepted.

Hence an ontology can be divided into three domains (see table below).

Table 1. Ontology domains (See [6])

Acronym	Domain	Names for Members
B	basis domain	bases, givens, basic elements or given elements
C	constructor domain	constructors
E	constructed domain	constructs or constructed elements

The basis and constructed domains combine to form a domain whose members are called elements. All three domains combine to form the ontology's universe, whose members are called items.

The core theory uses ontological principles to show that one can generate all the constructed elements (the constructed domain) from the basis and the constructor domains. As Fine notes, this means we do not need to use the constructed domain, E, to characterise an ontology, we can just use the couple < B, C>; E can then be generated from B and C. Though the order of analysis may well be the opposite; where one starts with the elements and works out what the constructors are and so the bases.

This generation of E relies upon an exhaustive application of the constructors; where anything that can be generated is generated. In our approach, we find it useful to 'construct' this process. We call this the ontology's ONTOGENESIS and characterise the ontology as the couple < B, C > plus ONTOGENESIS; so, in a sense, ONTOGENESIS stands in for E.

Fine's extended theory is about how ontologies fit into an ontological space; where this is a nonempty collection of ontologies that conforms to certain principles. It shows how, given ontologies in a space, similar ontologies with permutations of basis and constructor domains also exist in the space. We extend this to permutations of the individual constructors in the domain.

This provides a framework for an incremental sandbox approach to explaining and understanding an ontology. If one wishes to understand an ontology (the target), these principles allow one to initially pick out from the ontological space ontologies that contain just the basis domain or just a single constructor from the target ontology and examine these. One can then pick and examine richer combinations, seeing how these lead to richer structures, until one arrives at the target ontology.

2.2 Finean General Composition Framework

Using this general ontological framework, in [7] Fine sketches a general unified framework for composition; the ways in which one object can be a part of another.

It is distinctive in several ways (all of which suit our current purpose). It takes a very liberal notion of part that encompasses both traditional mereological relations as well as others, such as set-membership, that are not usually thought of as whole-part relations; in other words, for Fine set-membership in another kind of whole-part. (The term mereological whole-part will be reserved for the traditional mereological relations.) The formulation of the framework takes the operation of composition (and decomposition) as primitive rather than the more familiar relation of whole-part. These primitive operations are then treated as constructors within the general ontological framework; where each fundamental constructor generates a different kind of whole-part. As noted in Bennett [10], one key choice is the direction of generation for these composition constructors; whether parts generate wholes or the other way around. Standard mereology and cumulative set theory generate wholes from parts. Bennett offers Schaffer [11] as an example of choosing the other direction.

Fine formulates his framework in terms of compositional principles; which act like axioms. His first broad division is formal and material. Formal principles can be further divided into those that deal with conditions of application and those that provide identity conditions. Material principles provide, for example, conditions for the presence of a whole or part in space and time or at a world.

Fine develops an very simple way of characterising the formal identity principles for summative identity (identity for the mereological sum operation) based upon a notion of regular identity conditions (the reader can find the details in [7]). The result is the four CLAP principles in Table 2, so-called because of their initials:

Table 2. CLAP (formal identity) principles (See [7])

C	*Collapse*	$\sum(x) = x$	If Collapse holds then any whole composed of a single part is identical to it.
L	*Levelling*	$\sum(\dots ,\sum(x, y, z,...),\dots ,\sum(u, v, w,...),...) = \sum(\dots , x, y, z,\dots ,\dots , u, v, w,\dots , ...)$	If Levelling holds then when the parts of whole have parts, these parts' parts are also parts of the whole.
A	*Absorption*	$\sum(\dots , x, x, \dots , \dots , y, y, \dots , \dots ,) = \sum(\dots , x, \dots , y, ...)$	If Absorption holds then the repetition of parts is irrelevant to the identity of the whole.
P	*Permutation*	$\sum(x, y, z, ...) = \sum(y, z, x, ...)$ (and similarly for all other permutations)	If Permutation hold then the order of the parts is irrelevant to the identity of the whole.

A constructor's formal identity can be characterised by whether these principles or the counter-principles hold for it – one can summarise this into a CLAP profile, with a mnemonic where the appropriate letter is struck through when the counter-principle

holds. Its application is also characterised by its direction of generation. Table 3 gives examples of constructors for each CLAP profile and direction.

Table 3. Some possible forms of composition (See [7])

Profile	Whole	Example Parts-to-Wholes Constructor	Example Wholes-to-Parts Constructor
CLAP	Sums	SUM-BUILDER	SUM-DECOMPOSER
~~CL~~AP	Sets	SET-BUILDER	SET-DECOMPOSER
CLA~~P~~	Strings	STRING-BUILDER	STRING-DECOMPOSER
~~CL~~A~~P~~	Sequences	SEQUENCE-BUILDER	SEQUENCE-DECOMPOSER

The composition constructors construct elements – the constructed elements. They also map out the composition structure; how wholes are related to parts and by which kind of whole-part. However, not every application of a constructor constructs new elements; some of these non-generational constructions map new composition structure (others neither construct new elements or new mappings). Sum provides us with a good example of this. If we start with a molecule *abc* of three atoms as the only given, then a single sum-decomposition into all its parts will construct new elements; *a, b, c* as well as the molecules *ab*, *ac*, and *bc*. It also give us the composition mapping for *abc*; the relations between *abc* and all its parts. If we apply sum-decomposition to *ab* we get *a* and *b* – but these are not new, they are in a sense *re*-constructed as they were newly constructed in the first decomposition. However, it gives us a new composition mapping for *ab*, something that was not given in the first decomposition. It also shows, in some sense, *ab* is prior to *a* and *b.* To capture these characteristics, we call the initial types of construction *generative*. The second decomposition does some work, as it shows that that *ab* is composed of *a* and *b*, so we call this type of construction *compositional*. Earlier we introduced the process ONTOGENESIS, which exhausts the ontology's generative power. However, as the example shows, this may not exhaust the compositional power. To do this we extend ONTOGENESIS to cover all the possible compositional relations, and call this ONTOGENESIS+. This is needed to extract the full power of the constructors.

Generative Hierarchies. The generative power of constructors provides a way of organising elements into levels within hierarchies. Every constructed element is constructed by a sequence of generative constructions. This provides a simple way to organise them into a hierarchy: each level is characterised by the number of times the constructor has been applied. The additional compositional constructions are not considered here.

One can also see this as a form of ONTOGENESIS, where one is given a START collection and a constructor, and elements are constructed by repeatedly applying the constructor. One needs to be clear what the constructor is being repeatedly applied to. For this it is useful to introduce the notion of *stage* and to help define it the notion of *generation*. A generation is all the (new) elements generated at a level–this excludes cases where the element is re-constructed. And a stage is all the elements in a level's generation and all the prior levels' generations. Then a *stage level hierarchy* is generated by repeatedly applying the constructor to stages.

We can make the process a little more formal; we need to account for the cases where the START collection has multiple elements and there are multiple ways to choose the collections to which the constructor is applied. For this we define an operation POWER that takes a collection and selects all the possible sub-collections from this (in our example, we do not need to take account of sensitivity to duplications and permutations). In the case of sets, there is a connection with the set-theoretic powerset axiom. Then, given a constructor, one can create a stage level hierarchy in steps (using *constructor* and *start* variables) – where any element that can possibly be generated from the constructor will be generated at some level - as follows:

Generation 0: *start*
Generation 1: *constructor* (POWER (Stage 0))
…
Generation N+1: *constructor* (POWER (Stage N))

where N is unbounded and the constructor variable ranges over all the constructors (this is needed where there is more than one constructor). For this example, we do not need to consider actual infinite or transfinite recursion. Note that this hierarchy is mixed in the sense that its component parts can be from multiple levels.

There is a variant hierarchy based upon generations rather than stages - a *generation level hierarchy*. This is of interest because the multi-level hierarchies considered in some multi-level modelling seem to be of this type – see, for example, the 'level respecting' principle in [12] and the discussion of strict metamodelling in [13]. We look at this in Section 4.

A generation level hierarchy is one where the next level is generated by applying the constructor to the previous level's generation. Each step of the generation level hierarchy only considers the preceding level's generation and so ignores any earlier levels. This means it does not use the full generative power of the constructor. We can formalise this by replacing stage with generation in the earlier process as follows:

Generation N+1: *constructor* (POWER (Generation N))

Note that this hierarchy is pure (that is, unmixed) in the sense that its component parts at each level are from a single level.

2.3 Framework Clarifications

There are a few points that we clarify as scene-setting for the sandbox example; kinds and unique decomposition.

Constructed elements are linked to the constructor that constructs them; this can form the basis for kinds. SET-BUILDER is a good example. An element is of the kind *set* if and only if it is constructed by SET-BUILDER. One can also base kinds of whole-parts upon the constructor from which they emerge – so set membership is the kind of whole-part that emerges from SET-BUILDER; relating the (whole–set) constructs with the (parts–members) constructees. Kinds are useful in defining the application scope of the constructor operations; for example, SUM-BUILDER does not apply to elements of the kind *set* – nor does it construct elements of this kind.

In the case of sets, strings and sequences, there is a unique collection of parts that compose the wholes – the principles merely regulate the permutations and duplicates of the parts and the levelling of the constructors. In standard mereology, which is based upon the mereological whole-part relation, sums work in a different way. There are

multiple ways in which a whole can be a sum of its parts. Our framework starts with composition (and de-composition) rather than whole-part and this provides us with a way to ensure a unique decomposition. One can decompose a sum uniquely into a collection of parts. A good illustration of how this would work is a universe of mereological atoms. All the wholes that are sums of mereological atoms would be the sum of a unique collection of these atoms. If one did not want to assume mereological atoms, one can take the decomposition to be all the parts. We adopt this latter approach for the example.

3 Analysing a SIMPLE Sandbox Example

We now describe a simple example ontology, called (unsurprisingly) SIMPLE. The SIMPLE ontology is intended to illustrate how our approach is useful in explaining the formal requirements for metaphysical structures in models. Given this goal, we aim to make the example as simple as possible while still being able to illustrate how the structural whole-part patterns emerge, with a focus on the hierarchies and levels (such as generation and stage level hierarchies) within the patterns.

3.1 Building Up the SIMPLE Ontology

As noted earlier, in the Finean ontology framework, we can characterise an ontology in terms of its basis and constructor domains, which we now do. Our full example ontology, called SIMPLE, has a non-empty basis domain and a constructor domain containing two constructors (SET-BUILDER and SUM-DECOMPOSER).

Under our approach, based upon the Finean Extended Theory, we construct an ontological space with ontologies that take us in small incremental steps from the NULL ontology to the final SIMPLE ontology. This is essentially all the permutations of basis domain and individual constructors (as the table below shows). Some of the permutations are not illuminating, so are not visited in the analysis – these are marked in the table.

Table 4. SIMPLE's analysis ontological space

Ontology	Acronym	Basis Domain	SET-BUILDER	SUM-DECOMPOSER	Includes	Analysed
NULL	NULL	NO	NO	NO	None	NO
Simple Basis only	SB	YES	NO	NO	NULL	YES
SET-BUILDER only	SET	NO	YES	NO	NULL	YES
SUM-DECOMPOSER only	SUM	NO	NO	YES	NULL	NO
SET-BUILDER and SUM-DECOMPOSER	SET+SUM	NO	YES	YES	SET, SUM	NO
Simple Basis plus SET-BUILDER	SB+SET	YES	YES	NO	SB, SET	NO
Simple Basis plus SUM-DECOMPOSER	SB+SUM	YES	NO	YES	SB, SUM	YES

SIMPLE	SIMPLE	YES	YES	YES	SB+SET, SB+SUM, SET+SUM	YES

One could regard the other ontologies as partial versions of the SIMPLE ontology, or subontologies of it. We start the analysis with the Simple Basis Only (Sub-)Ontology.

Simple Basis Only (SB) Ontology. From a metaphysical viewpoint, the choice of bases typically involves important architectural commitments [8]. There are a variety of options. One could start with a basis domain of mereological atoms [14] and build up the ontology from them. Or one could adopt priority monism [11], then the given will be everything (which may be a single actual or an infinity of possible worlds) and the ontology is built by decomposing this. Choosing one of these would then dictate the direction of the intended SUM constructor; whether to start with mereological parts and construct wholes – or vice versa, start with a mereological whole and construct the parts.

To keep things simple, we follow the priority monism route and have a basis domain consisting of a single object – a pluriverse of possible worlds [15], which we will abbreviate as PV. Again, to keep things simple, we adopt super-substantivalism [16], which [17] calls monistic substantivalism; this considers matter to be identical to the spacetime region is occupies.

We can now define the ontology as SB = < (PV), () >. As there are no constructors in this ontology then ONTOGENESIS is the NULL process – and PV is the only element (and item) in the ontology. From a hierarchy perspective, this is a limit case. There is a single object which can be regarded (pathologically) as a stage level hierarchy.

SET-BUILDER Only (SET) Ontology. This ontology has an empty basis domain and a single constructor SET-BUILDER, mentioned earlier, so SET = < (), (SET-BUILDER) >. This constructor is introduced in [6] and described in detail in [7]. We mentioned SET-BUILDER earlier in Table 4, noting it is formally a SET constructor where the direction of generation is part to whole and its form of identity is ~~CL~~AP; it works as follows:

- It has the associated kind, *sets*. An element is a set if and only if it is constructed by SET-BUILDER.
- It is presented with a collection (possibly empty) of elements (parts) and it constructs a set (the whole).
- If presented with a collection of zero elements it generates the empty set, {}.
- Its ~~CL~~AP profile means that if presented with non-zero collection of elements it generates the set of those elements, ignoring duplicates and order.

This provides us with sufficient resources to develop a good example of generation and stage level hierarchies. Using the process schema defined earlier, we can create SET's generation level hierarchy in steps as follows:

Generation 0: ()
Generation 1: SET-BUILDER (POWER (Generation 0))

…
Generation N+1: SET-BUILDER (POWER (Generation N))

The first four levels' generations and stages are shown below.

Table 5. SET-BUILDER generation level hierarchy

Level	0	1	2	3	4
Generation		{}	{{}}	{{{}}}	{{{{}}}}
Stage		{} +	{{}} + {} +	{{{}}} + {{}} + {} +	{{{{}}}} + {{{}}} + {{}} + {} +

This provides us with an example of the point made earlier, that the generation level hierarchy is not necessarily the whole constructed domain as, at each level, the constructor is only applied to the previous generation. For example, at stage 2, the universe contains both generation 1 and 2 elements, but SET-BUILDER is only applied to the generation 2 elements.

Of course, we could take a related constructor GENERATION-SET-BUILDER whose application is restricted to generation collections of elements, then the hierarchy would cover the domain. However, we would then have to metaphysically justify the choice of this constructor.

We can reinforce this incompleteness point by generating the corresponding stage level hierarchy by replacing generation with stage as noted earlier, giving:

Generation N+1: SET-BUILDER (POWER (Stage N))

The first three levels are shown in the table below.

Table 6. SET-BUILDER stage level hierarchy

Level	0	1	2	3
Generation		{}	{{}}	{{{}}}, {{}, {{}}}
Stage		{} +	{{}} + {} +	{{{}}}, {{}, {{}}} + {{}} + {} +

The full process exhausts the generative power of the basis domain and constructors and so; **ONTOGENESIS (SET):** SET-BUILDER Stage Level Hierarchy.

Every application of the SET-BUILDER is generative, so there is no difference between ONTOGENESIS (SET) and ONTOGENESIS+ (SET).

Simple Basis plus SUM-DECOMPOSER (SB+SUM) Ontology. This ontology is an extension of the Simple Basis Only Ontology with the SUM-DECOMPOSER constructor, so; SB+SUM = < (PV), (SUM-DECOMPOSER) >. We see from Table 4, that SUM-DECOMPOSER is a SUM constructor where the direction of generation is whole to part and its form of identity is CLAP. It works as follows:

- It has the associated kind, material elements. PV is a material element, all the elements constructed by SUM-DECOMPOSER are also material elements and these are the only material elements.
- It is presented with a single material element (the whole) and it constructs a collection of material elements (parts).
- Its CLAP profile means that it generates a collection of elements with no duplicates or order.

The choice of this constructor naturally complements the choice of PV as a basis. Given the super-substantival choice for PV, SUM-DECOMPOSER takes a material element and uniquely decomposes it into all its material, spatiotemporal parts. One can see the constructor as establishing the parts of which the whole is composed.

This ontology has a simple two level hierarchy, with PV as generation 0 and all its parts as generation 1. And only one application of the constructor gives all the constructed elements; the parts of the whole PV. So: **ONTOGENESIS (SB+SUM):** SUM-DECOMPOSER (PV). ONTOGENESIS+ (SB+SUM) needs to consider these parts as wholes and establish their parts, and this is achieved by applying sum-decomposer to each of them. We specify this process using an iterative FOR EACH component process – as follows:

Generation 0: (PV)
Generation 1: SUM-DECOMPOSER (PV)
Generation 2: FOR EACH X in Generation 1 (SUM-DECOMPOSER (X))

The two generations exhaust the compositional power of the constructor.

The SIMPLE Ontology. This ontology is Simple Basis plus SET-BUILDER and SUM-DECOMPOSER, so SIMPLE = < (PV), (SET-BUILDER, SUM-DECOMPOSER) >. It is a combination of all of the earlier ontologies, where the basis domain and constructors were defined and much of the analysis done.

This is the first ontology with multiple constructors. For the single constructor situations, we had two simple levelling schemes based upon the number of applications of the constructor; we combine these for multiple constructors to give us ONTOGENESIS, as follows:

Generation 0: (PV)
Generation 1: SUM-DECOMPOSER (PV) + SET-BUILDER (POWER (PV))
Generation 2: SET-BUILDER (POWER (Stage 1))
…
Generation N+1: SET-BUILDER (POWER (Stage N))

And we build ONTOGENESIS+ (the full composition map) in the same way as in SB+SUM adding this to generation 2:

Generation 2+: FOR EACH X in Generation 1 (SUM-DECOMPOSER (X))

The ONTOGENESIS hierarchies do not have the same symmetry as the cumulative set hierarchy. For example, the standard cumulative set hierarchy all the singleton sets are at the same level (see SET above). However, as shown in the table below (where p1, p2, … are the parts of PV), the combination of the two constructors produce, at level 1, not only all the parts of PV but also the empty set and singleton PV – and produces at level 2 singletons of the empty set and the parts of PV as well as singleton-singleton PV.

Table 7. Example multiple constructor hierarchy

Level	0	1	2
Generation	PV	{}, {PV}, p1, p2, …	{{}}, {{PV}}, {p1}, {p2}, …

This suggests another way to construct hierarchies; by calculating the levels using a single constructor. In this case, by only considering the generative applications of SET-BUILDER– ignoring SUM-DECOMPOSER. This recaptures the symmetrical hierarchy.

4 Deriving Structures From SIMPLE

The example shows how fundamental multi-level hierarchies emerge from the pattern of construction. However, this is not the only way that these hierarchies, multi-level or otherwise, can emerge. The fundamental structures can be used to derive other types of compositions and their associated hierarchies.

4.1 Deriving SIMPLE's Subset Constructor

One of these derived types of composition is subset or set-inclusion. For our purposes, it makes sense to do this using a derived constructor SUBSET-DECOMPOSER (*set*) that works as follows:

- Its direction of generation is whole to parts.
- Its form of identity (like SUM-DECOMPOSER) is CLAP, which means that it generates a collection of elements with no duplicates or order.
- It is presented with a single set element (the whole) and it constructs a collection of set elements (the subset parts) – hence it can only be applied to elements of kind *set* and it constructs elements of the same kind.

Applying this constructor to a set will produce a collection of all its subsets. This can be regarded as the initial stage in the set-theoretic power set axiom. To formalise this, we specify the inverse of SET-BUILDER; SET-DECOMPOSER (*set*) which takes a set and produces a collection of its members. Using this we can specify; SUBSET-DECOMPOSER (*set*) = SET-BUILDER (POWER (SET-DECOMPOSER (*set*))). This, when given a set takes its members and constructs sets from them, constructing all its subsets. As with SUM-DECOMPOSER, one then needs to apply the same operation to each of the parts to establish the full subset composition mapping, as shown below.

Generation 0: (*set*)
Generation 1: SUBSET-DECOMPOSER (*set*)
Generation 2: FOR EACH X in Generation 1 (SUBSET-DECOMPOSER (X))

One can begin to see the structural similarities between SUBSET-DECOMPOSER and ordinary SUM-DECOMPOSER noted in [18]. For example, like the SUM-DECOMPOSER constructor, the SUBSET-DECOMPOSER constructor is not levelled (in CLAP terms) and so has a flat structure. However, one can also see, as [7] notes, that in this space the first is fundamental and the other is derived.

4.2 Deriving SIMPLE Taxonomies and Component Breakdowns

One can derive a new hierarchy by subsumption from a whole-part hierarchy – and this can be levelled in ways that the original hierarchy is not. Taxonomies, including ones such as the Linnaean classification, and component breakdowns, in so far as they are ontological, are good examples. The procedure is simple. One chooses a subset of the elements and then a subset of their whole-part compositions so that the result conforms to the desired whole-part structure; if one wants a levelled structure, then one ensures the structure conforms to the right CLAP principles – typically that it does not adhere to levelling.

Consider (one version of) the Linnaean classification hierarchy that has ‘Natural Things’ at the top and is divided and sub-divided through the levels until reaching ‘Felis leo’ and ‘Felis tigris’ as the bottom. Within the SIMPLE example, ‘Natural Things’ is a set of (spatiotemporally extended) material elements and the other classifications are subsets of it [5]. This collection of classifications are the elements used in the hierarchy. These can be derived using SUBSET-DECOMPOSER and a filter, FILTER-LC, that given a collection selects those elements that are Linnaean classifications; **LINNAEAN-C:** FILTER-LC (SUBSET-DECOMPOSER (Natural Things)).

Traditionally, the classification structure is levelled by taking a transitive reduction of the underlying whole-part hierarchy. We can derive the LINNAEAN-SUBSET-DECOMPOSER constructor for this by taking a constrained version of SUBSET-DECOMPOSER that can only be applied to Linnaean classifications and when applied only reruns the next level – other levels are filtered out. This gives us a natural generation level hierarchy LC, which is defined as having ‘Natural Things’ as its base and LINNAEAN-SUBSET-DECOMPOSER as its sole constructor – and;

Generation 0: (Natural Things)
Generation 1: LINNAEAN-SUBSET-DECOMPOSER (Natural Things)
Generation 2: FOR EACH X in Generation 1 (LINNAEAN-SUBSET-DECOMPOSER (X))
…
Generation N+1: FOR EACH X in Generation N (LINNAEAN-SUBSET-DECOMPOSER (X)),

Of course, the classification structure is richer; which means more constructors are needed. For example, the levels (e.g. Kingdoms) are explicit as ranks (this is described in [5]), so a RANKER constructor is required. However, this skeleton outline should be sufficient to indicate how this could be done.

The same derivation process can be used on other kinds of whole-parts. Component breakdown structures (such as car X breaks down into body and engine components and engine into so on) would be based upon mereological whole-part and SUM-DECOMPOSER [19].

This ability to derive levelled hierarchies from the fundamental structures not only helps to explain (ontologically) what these hierarchies are but also suggests there may be new areas for MLM tools to be deployed.

4.3 Deriving MLM Generalisation and Classification Hierarchies

Analyses of classification and generalisation have noted the similarities with, respectively, set-membership and subset [3]. A common point made is that generalisation (like subset) is transitive and does not have levels whereas classification (like set-membership) is anti-transitive and has levels [12]. As noted earlier, the sandbox analysis both captures these formal structures and exposes the close relationship between the two (when seen as compositions based upon constructors); that SUBSET-DECOMPOSER is derived from the fundamental SET-BUILDER.

However, classification is not plain set-membership nor generalisation plain subset – and the differences can guide us on how to build the appropriate derived constructors. One key difference is that the conceptual models typically restrict their relations to a sub-domain of interest. As with the taxonomies above, this can be captured by a filter on the constructor – where the filter clearly delineates the scope of the domain.

In UML and MLM there are many varieties of generalisation. In some MLM contexts, the generalisation hierarchy is, like in taxonomies above, restricted to a transitive reduction. This is also common in conceptual modelling contexts, where typically only the transitive reduction is modelled, and the transitive closure is rarely if ever mandatory. In some contexts, the hierarchy is restricted to a tree-structure, whereas in others it is more usual to allow lattice (multiple inheritance) hierarchies. Submitting these different structures to a sandbox analysis would capture their different commitments in derived constructors (typically using filters) with their appropriate CLAP profile – as well as their common underpinnings.

Similarly, there are varieties of classification. As noted earlier, [12] introduces the property of 'level respecting' in a way that leads to a similar structure to generation hierarchies. The appropriate constructor for this shape of structure, GENERATION-SET-BUILDER, is similar to SET-BUILDER, but with a filter on what it applies to. This analysis of the structures in terms of how they are derived from more fundamental constructors helps to both formalise their differences as well highlighting them.

It also gives rise to natural enquires as the motivations for the choices; for example, are they adopted for metaphysical/ontological or pragmatic reasons, and if so, what are these reasons? So, for example, it makes clear that adopting a GENERATED-SET-BUILDER (as strict metamodelling does) filters out mixed sets. One can ask what the motivation for this is and whether the cost of excluding them is worthwhile.

5 Conclusions

We have provided an outline of the underlying framework upon which the ontological sandbox is built and indicated where further details can be found [6, 7]. We have used the example SIMPLE ontology to show how one can use this sandbox to build up an understanding of the target ontology in steps through ontological space. We have shown how the constructional approach exposes the underlying compositional metaphysical structures of ontologies; for example, how, in the SIMPLE ontology at least, the set-inclusion hierarchy is derived from the fundamental set-membership hierarchy's constructor. We have shown how understanding the CLAP principles of composition can expose the architectural choices made and suggest alternatives. We have shown how familiar hierarchical structures, such as taxonomies and component breakdowns, can be derived from more fundamental structures. We have shown how the same techniques can be applied to MLM classification and generalisation structures. We have provided a clear picture of the trade-off between order and expressiveness that drives the choice of strict metamodelling (effectively a generation level hierarchy) – as well as an alternative – stage level hierarchy. Hopefully, this is sufficient to provide a good idea of the potential for this sandbox to help us understand and improve the ontological underpinnings of conceptual models.

This paper provides an outline of the ontology sandbox. A lot more work needs to be done showing how it can be used. One area where we think it will be fruitful to develop the approach is investigating the range of possible ontological commitments of existing MLM frameworks and tools. This could not only suggest possible ontologies

(and so semantics) for the frameworks but also identify possible ontologically-driven improvements.

References

1. Atkinson, C., Gerbig, R., Kühne, T.: Comparing multi-level modeling approaches. MULTI@ MoDELS. pp. 53–61 (2014).
2. Atkinson, C., Kühne, T.: Model-driven development: a metamodeling foundation. Software, IEEE. 20, 36–41 (2003).
3. Frank, U.: Thoughts on classification/instantiation and generalisation/specialisation. (2012).
4. Kühne, T.: Contrasting classification with generalisation. Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96. pp. 71–78. Australian Computer Society, Inc., Darlinghurst, Australia (2009).
5. Partridge, C., de Cesare, S., Mitchell, A., Odell, J.: Formalization of the classification pattern: survey of classification modeling in information systems engineering. Software & Systems Modeling. 1–37 (2016).
6. Fine, K.: The study of ontology. Noûs. 25, 263–294 (1991).
7. Fine, K.: Towards a theory of part. The Journal of Philosophy. 107, 559–589 (2010).
8. Partridge, C.: Note: A couple of meta-ontological choices for ontological architectures. LADSEB CNR, Padova, Italy. (2002).
9. Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Extending the foundations of ontology-based conceptual modeling with a multi-level theory. International Conference on Conceptual Modeling. pp. 119–133 (2015).
10. Bennett, K.: Construction area (no hard hat required). Philosophical Studies. 154, 79–104 (2011).
11. Schaffer, J.: Monism: The priority of the whole. Philosophical Review. 119, 31–76 (2010).
12. Kühne, T.: Matters of (meta-) modeling. Software and Systems Modeling. 5, 369–385 (2006).
13. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. ACM Transactions on Modeling and Computer Simulation (TOMACS). 12, 290–321 (2002).
14. Goodman, N.: On relations that generate. Philosophical Studies. 9, 65–66 (1958).
15. Lewis, D.: On the plurality of worlds. Blackwell, Oxford (1986)
16. Sklar, L.: Space, time, and spacetime. Univ of California Press (1977).
17. Schaffer, J.: On What Grounds What. In: David Manley; David J. Chalmers; Ryan Wasserman (ed.) Metametaphysics: new essays on the foundations of ontology. pp. 347–383. Oxford University Press (2009).
18. Lewis, D.: Parts of classes. B. Blackwell (1991).
19. Sanfilippo, E.M., Masolo, C., Borgo, S., Porello, D.: Features and Components in Product Models. FOIS. pp. 227–240 (2016).