The Information Management Framework

Core Constructional Ontology

The Foundation for the Top-Level Ontology of the Information Management Framework

Version 1.91

DRAFT

April 6, 2022

Contents

Executive summary **4**

Introduction **4**
Background 4
Purpose 5
Target Audience 5

1 Context **5**

2 Report overview **6**

3 Project background **7**

4 Constructional ontology **9**

5 Core Constructional Ontology **10**

6 Developing the Core Constructional Theory **11**

7 Technical background **13**

8 Core Constructional Theory: the formal language **14**
8.1 Logical framework 14
8.2 Non-logical vocabulary 16
8.3 Conventions 18

9 Core Constructional Theory: the axioms **19**
9.1 Plural logic 19
9.2 Stages 21
9.3 Initial stage 23
9.4 What exists at stages 24
9.5 What is and is not constructible 26
9.6 Generic constructor 27
9.7 Specialised constructors 27
9.8 Classification 28
9.9 Set constructor 29
9.10 Sum constructor 31
9.11 Left and right constructors 34
9.12 Pair constructor 36
9.13 Union constructor 38
9.14 Traces 39
9.15 Maximal extension of a stage 42
9.16 Induction on the construction of objects 43

10 Derivation of set theory and mereology **43**
10.1 Axioms of set theory 43
10.2 Derivation of the axioms of set theory 46
10.3 Axioms of mereology 48
10.4 Derivation of the axioms of mereology 49

11 Consistency of the Core Constructional Theory **50**

12 Conclusion **51**

A Notions **53**
A.1 List of key types and identity criteria 53
A.2 Constraints on inputs 54

B Design choices **54**

C Supporting the IMF's selected TLOs **56**

D CLAP background **58**

E Constructing via CLAP profiles **61**
E.1 Induction on the construction of Ks 62
E.2 From sufficient to necessary conditions for identity 63
E.3 An intended model of our construction 66
E.4 Equivalent formulations of the extremal clause 67
E.5 Further investigations 71

F Proof of consistency of the Core Constructional Theory **71**

G Axioms **77**
G.1 Primary axioms 77
G.2 Optional axioms 84

H Future work **84**
H.1 Short term 85
H.2 Long term 85

I Literature sources **85**
I.1 Current situation with foundations 85
I.2 History of constructional ontology 86
I.3 Current work on constructional ontology 87
I.4 Other background work 87

References **88**

Acknowledgments **91**

Executive summary

West forthcoming describes the seven-circles approach that is being used to organise the development of the Information Management Framework. The focus of this report is on the seventh circle: the Core Constructional Ontology.

An investigation (West 2020) recommended that the top-level ontology of the Information Management Framework be underpinned by rigorously established foundations. This report develops a constructional approach to ontology, the Core Constructional Ontology, that provides these foundations. The report describes the ontology and its motivation, and it provides its first formalisation, thereby giving the Core Constructional Ontology a rigorous foundation. The formalisation is proved to be consistent.

With these in place, we have established the feasibility of formalising the foundation of the Information Management Framework's top-level ontology and provided a base for the building of the top-level ontology. Future work will refine this formalisation.

Introduction

Background

In 2017, the National Infrastructure Commission published "Data for the Public Good" (NIC 2017) which sets out a number of recommendations including the development of a UK National Digital Twin supported by an Information Management Framework of standards for sharing infrastructure data, under the guidance of a Digital Framework Task Group set up by the Centre for Digital Built Britain.

Much work has been done following this, but in particular:

- A vision of how society can benefit from a UK National Digital Twin is set out in "Flourishing Systems - Re-envisioning infrastructure as a platform for human flourishing" (Burgess et al. 2020).
- The direction for the technical standards, guidance and common resources needed as part of the Information Management Framework is set out in "The pathway towards an Information Management Framework - A 'Commons' for Digital Built Britain" (Hetherington and West 2020).

In particular the latter identified the need for:

- A **Foundation Data Model**: a data model that provides the structure and meaning of data incorporating a top-level ontology based on science and engineering principles, enabling it to be extended to support the broadest possible scope consistently.
- A **Reference Data Library**: the classes and properties needed to enable different organizations and sectors to describe things consistently.

- An **Integration Architecture**: the technical means, including open source software, for sharing data securely with authorised users.

It also set out an outline approach and plan to develop the Information Management Framework.

Purpose

The purpose of this report is to give an understanding of the technicalities of the foundation and formalisation underpinning the Information Management Framework's ontology.

Target Audience

This report is directed at a technical audience interested in understanding what the foundation of the Information Management Framework's ontology is and how it is formalised. In particular, we expect the report to be of interest to logicians and formal ontologists.

1 Context

The National Infrastructure Commission's report "Data for the Public Good" (NIC 2017) recommended the creation of a National Digital Twin (NDT) connecting digital twins across different sectors to give a system of systems view of national infrastructure. As emphasised in reports such as Burgess et al. 2020, this harnessing of the power of information and data will deliver better decisions which lead to better outcomes for people and society.

The NDT is supported by an Information Management Framework (IMF) that includes a Foundation Data Model (FDM) as a key component (Hetherington and West 2020). An investigation (West 2020) recommended that the FDM seed be founded on a top-level ontology based on the four 4-dimensionalist top-level ontologies (TLOs) that it had been determined best met the technical requirements of the FDM, underpinned by rigorously established foundations.

Existing TLOs, including those selected as a starting point for the FDM, are typically assembled from disconnected core components. In the case of the selected TLOs, such components correspond to sets, parts (mereological sums), and relations, which were not built and formalised in a single, unified way. Previous work (De Cesare and Partridge 2016; Partridge, Cesare, et al. 2017; Partridge, Mitchell, Loneragan, et al. 2019; Partridge, Mitchell, Loneragan, et al. manuscript) has investigated how the selected TLOs could be unified based on a constructional framework developed by the philosopher and logician Kit Fine. Here we develop this novel approach, formally refactoring the disconnected core components into a unified theory. Using this unified theory to underpin the IMF's TLO will significantly simplify and strengthen it, as well as providing a firmer foundation for future development.

The National Digital Twin programme (NDTp) set up a project to build this unified ontology, called the Core Constructional Ontology (CCO). This stage of the project has developed a transitional framework that establishes the feasibility of building the CCO. The framework is formalised by means of a theory we call the Core Constructional Theory (CCT). Here we describes the CCT and its associated CCO. Later stages of the project will further develop and enhance this framework. Appendix E.5 gives some indication of what these enhancements could be.

This novel theory develops the idea that *all* the objects in the CCO emerge during construction. We start from an initial collection of objects—often called *givens*—and a small number of constructors, and the entire ontology unfolds from repeated constructions. So from the givens and constructors one knows, in principle, all the objects in the ontology. Using the technical resources of plural logic, the CCT formalises the arrangement of constructions in stages, where the intended ontology arises after exhausting all the stages. This report documents the CCT and provides a proof of its consistency.

2 Report overview

The report broadly divides into two parts. The first part is introductory and is accessible to non-specialist readers. The second part is more technical, as it describes the details of the formalisation. There are also various appendices.

The first, introductory part comprises five sections.

- Section 3 (Project background) provides the context for our project, its connection to the IMF, and its relation to TLOs.
- Section 4 (Constructional ontology) introduces the idea of constructional ontology.
- Section 5 (Core Constructional Ontology) describes the main features and benefits of the CCO, our own constructional approach.
- Section 6 (Developing the Core Constructional Theory) develops the basic framework for our formalisation of the CCO.
- Section 7 (Technical background) explains the level of specialist knowledge presupposed by different sections of the report. It also provides references for readers interested in acquiring the appropriate background for the relevant sections.

The second, more technical part also comprises four sections.

- Section 8 (Core Constructional Theory: the formal language) introduces the formal language in which the axioms of our theory will be formulated.
- Section 9 (Core Constructional Theory: the axioms) presents and explains the axioms of the CCT.

- Section 10 (Derivation of set theory and mereology) shows how set theory and mereology can be recovered from the CCT.
- Section 11 (Consistency of the Core Constructional Theory) provides the context for our proof of consistency of the CCT.

The main body of the report refers to appendices that contain a variety of supporting material. The key ones are:

- an explanation of the design choices we have made and their motivations (Appendix B, Design choices);
- a mathematical investigation of the inductive features of construction (Appendix E, Constructing via CLAP profiles);
- a proof of consistency for the CCT (Appendix F, Proof of consistency of the Core Constructional Theory);
- a description of future work (Appendix H, Future work).

In this report, we aim to use consistently technical terminology that either follows standard mathematical and philosophical usage or reflects usage in the survey Partridge, Mitchell, Cook, et al. 2020 and in relevant TLOs. For example, we will consistently write about 'elements' of sets and 'members' of pluralities, though these are used interchangeably in the literature.

3 Project background

The UK has initiated a programme to build a National Digital Twin: the National Digital Twin programme (NIC 2017 and Bolton et al. 2018). Hetherington and West 2020 recommends the adoption of an Information Management Framework that includes a Foundation Data Model (FDM) as a key component. West forthcoming describes the seven-circles approach that is being used to organise the development of the Information Management Framework (see Figure 1). The present report is part of the output of the Information Management Framework. Its focus is on the last, seventh circle: the Core Constructional Ontology.

It was proposed that the FDM be built on a TLO (Hetherington and West 2020). West 2020 determined that four selected 4-dimensionalist TLOs—the target TLOs—best met the technical requirements of the FDM. It recommended that these target TLOs should be underpinned by rigorously established foundations. The formalisation in this report provides the rigorous foundations for these target TLOs.

The NDTp has carried out a comprehensive survey of available TLOs (Partridge, Mitchell, Cook, et al. 2020), which also develops a framework for assessing ontologies, including TLOs. One dimension of this assessment, called "the vertical aspect", concerns the formal structure of the "core of basic ontological hierarchical relations that are typically found in top-level ontologies;

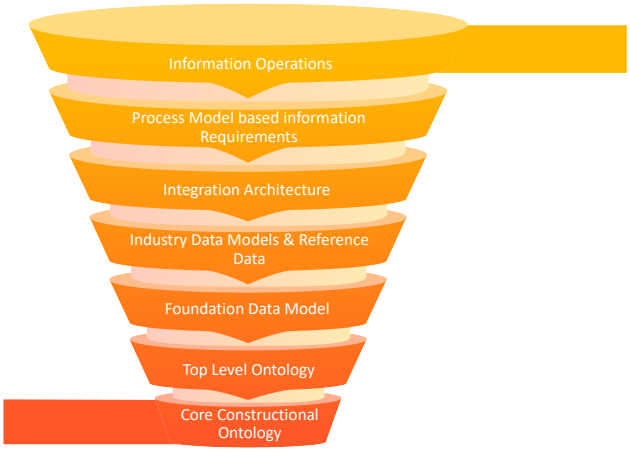

Figure 1: Information Management Framework: seven-circles approach

whole-part, type-instance and super-sub-type [...]. [I]n practice these hierarchies are a key part of the backbone of the ontological architecture" (Section 4.2.1). In the target TLOs, these are: whole-part, type-instance, tuple-place, and super-sub-type.

As noted in this survey, TLOs tend to use a number of well-known formalisations for the core hierarchical relations in the vertical aspect. The whole-part hierarchy is usually formalised using General Extensional Mereology (GEM), also known as Classical Extensional Mereology (see Varzi 2019 and Partridge, Mitchell, Cook, et al. 2020, Section 4.3.2). The target TLOs follow this approach. Some TLOs, including the target TLOs, adopt an extensional criterion of identity for types, that is, sets (described in Partridge, Mitchell, Cook, et al. 2020, Section 4.3.6). In these TLOs, the type-instance hierarchy is often formalised using a version of set theory that includes urelements and is also used to formalise the super-sub-type hierarchy. However, these TLOs adopt mereology and set theory as separate formal theories, without an integrated development of the two.

As Section 4.2.1 of the survey notes, the basic ontological hierarchical relations may be viewed as belonging to a single family and so should have similar formalisations. We remarked above on previous work that included initial investigations of this suggestion for the target TLOs. These investigations started to develop, but not yet formalise, the unified constructional approach sketched by Fine (most prominently in Fine 1991 and Fine 1999). The investigations

also noted that, for the unified approach, three basic forms of construction are required: one for sets, one for sums, and another for tuples, with an additional, derived constructor for subsets.

Our project was tasked with building upon this initial investigation to produce a unified formalisation of the core hierarchies required for the target TLOs. In particular, it was tasked with proving the consistency of the resulting formalisation.

4 Constructional ontology

In this section, we provide more context about the conception of ontology underlying our project, namely, constructional ontology. On this conception of ontology, one starts with some constructors and some givens. Then new objects emerge by construction, that is, from the application of constructors to objects. So the resulting ontology can be characterised by three parameters: the givens with which one starts, the constructors one employs, and the objects that emerge from applications of the constructors. Among the latter, we allow both constructed objects—the direct outputs of the construction—and traces of the construction process itself. (More details are given in later sections)

Two features of constructional ontology are worth highlighting here. First, it gives us a very clear picture of the overall contents of the ontology, including a comprehensive view of the categories of objects. For example, if the ontology includes only a set constructor, then all the objects in the ontology fall under one of these three kinds: givens, sets, and traces of the construction process. We start with the givens. Then sets and associated traces arise from repeated applications of the set constructor. In short, a constructor determines the category of the objects it generates.

Second, identity criteria emerge from the constructional process: the identity of constructed objects is dictated by their constructors and the inputs of the constructions. For examples, two sets (i.e. objects obtained from the set constructor) are identical if and only if they are constructed from the same objects. In short, a constructor determines the identity of the objects it generates as well as their category.

Constructional ontology has a long and venerable history (see Appendix I.2 for basic references). Of special importance for our project is Kurt Gödel's articulation of "the concept of set [...] according to which a set is anything obtainable from the integers (or some other well-defined objects) by iterated application of the operation 'set of' [...]" (Gödel 1964, p. 180). Here Gödel is referring to a broadly constructional concept of set: sets are generated by applications of the set constructor. More recent work on constructional ontology includes that of Kit Fine, who has significant influence on this project. We build on a number of his ideas for developing a unified constructional ontology.

5 Core Constructional Ontology

As we have mentioned, a constructional ontology can be characterised by three parameters: the givens with which one starts, the constructors one employs, and the constructed objects that emerge from applications of the constructors. We begin to develop our approach, the CCO, by specifying these parameters.

The CCO assumes that the givens are the spatiotemporal atomic elements of all possible worlds—sometimes called the *pluriverse*. In other words, these atoms "cover" the pluriverse. The theory developed in this report, the CCT, has a more flexible structure: the choices of givens is left open. More specifically, the CCT only requires a minimal collection of givens: those that are necessary to characterise the construction process (see Section 9.3). This simplifies the task of formalisation, where such a minimal collection is used. But the theory is consistent with stronger assumptions about the givens in the CCO and can be adapted straightforwardly to other settings.

Our approach involves a single, generic pattern for constructors, which we specialise into three key constructors: set, sum, and ordered pair. This specialisation involves characterising the conditions under which these constructors can be applied.

From these emerge the key categories of constructed objects: sets, sums, and ordered pairs. (Hereafter we refer to ordered pairs simply as *pairs*, since unordered pairs do not play any important role in this context.) Note that, as new objects emerge, more possibilities for construction become available. For example, once two sets a and b have been constructed, we can construct their singletons or the set that has just these two objects as elements. We generate the target ontology after we exhaust all possible constructions based on the generic constructor.

In the remainder of this section, we focus on three features of our approach and explain their benefits for our project: the approach is foundational, it has a high degree of unification, and it is constructional in the sense introduced above.

Our ontology has three characteristics that enable it to play a foundational role. The first is its *categorical completeness*. That is, the ontology provides the three basic categories of objects for the target TLOs: sets, sums, and tuples, together with their associated hierarchical relations (type-instance, super-subtype, whole-part, and tuple-place). The second characteristic is *object completeness*. That is, the ontology generates all the objects needed by the target TLOs. One might think of it as an "object factory" which supplies the objects that might be needed in any domain. The third characteristic is that the basic categories provide objects in the ontology with appropriate *identity criteria*. These are broadly extensional, based on the type of constructor and its input. For example, two sets are identical if and only if they are constructed from the same objects. Similarly, two sums are identical if and only if they have the same parts.

Our approach is unified in the following ways. First, it gives a *common development of three key domains*: sets, sums, and tuples. Ontologies that in-

volve sets and sum usually adopt set theory and mereology as separate theories, without an integrated development. Here we provide a unified treatment of sets, sums and pairs (hence tuples) as *sui generis* objects. So the three target domains—sets, sums, and tuples—arise in similar ways through construction. Second, there is a *common basis for identity criteria*, which are crucial for the foundational role discussed above. Identity criteria for the objects of the basic types are extensional, with differences arising from the way the objects are constructed. Third, the approach offers *uniform ways of capturing key commonalities and differences among objects of the basic types*. Such commonalities and differences are captured by features of the underlying constructors.

In the previous section, we outlined the basic structure of a constructional ontology. We now want to highlight four of its main benefits.

- *Categorical differences are constructional differences*: in constructional approaches, different kinds of objects can be distinguished from one another by the manner in which they are constructed. So categorical differences are explained by constructional differences.
- *Dependency*: some objects are built from others and hence "depend on" them. This provides an explanation of ontological dependence and the associated notion of grounding.
- *Reduction*: the ontology is built out of a relatively small set of initial objects and thus achieves fundamental ontological parsimony in the sense of Schaffer 2015.
- *Consistency*: construction can be a basis for consistency, avoiding paradoxes such as those of Russell and Burali-Forti, although we need to take care with the construction process. In our case, this is achieved by requiring that the construction be "bottom-up" in the sense that the properties of the constructed objects are determined by the properties of the inputs to the construction.

Gödel stresses this last benefit in the article cited above. Speaking of the constructional view of sets, he says:

> [It] has never led to any antinomy whatsoever; that is, the perfectly 'naïve' and uncritical working with this concept of set has so far proved completely self-consistent. (Gödel 1964, p. 180)

In this report, we show that our approach can be developed consistently. We do so by providing a mathematical proof of consistency for the associated formalisation.

6 Developing the Core Constructional Theory

The project was tasked with developing the CCT, a formalisation of the CCO. The first decision concerned what kind of constructional framework to adopt. It

was essential to demonstrate that this innovative approach would work, so we decided to go with the established, and thus safe, option of adopting a stage-theoretic framework (described below). This framework is well-developed for set theory but needed to be extended to sums and tuples. We also recognised that plurals have a natural role in representing the multiplicity of the input to our constructors. (For example, it is far more natural to describe the set $\{a, b\}$ as constructed from the *elements* a and b than from any single entity representing both a and b.) These things together provided us with the overall framework for our formalisation.

Our formalisation takes its inspiration from the influential iterative conception of sets, which goes back to Gödel's article cited earlier: "What is Cantor's Continuum Problem?" (1964). According to this conception, the sets are formed in stages. We start, at stage 0, with some urelements. At stage 1, we form all sets of urelements, which results in a larger domain. At stage 2, we form all sets of objects from this larger domain. We continue into the infinite. At limit stages, such as the first infinite stage ω, we collect the previously formed sets, which are then used to form new sets at the next stage $\omega + 1$. The most famous development of the iterative conception of sets is due to George Boolos (Boolos 1971), who develops a stage theory—an axiomatic theory that describes how sets are successively formed at stages.

We generalise and extend Boolos's stage theory in three different ways. First, instead of considering just the construction (or "formation") of sets, we adopt Fine's framework and offer a unified account of three different kinds of construction that are central to the target TLOs: sets, mereological sums, and pairs. This enables us to capture the core hierarchical relations of the target TLOs mentioned earlier, namely, type-instance, super-sub-type, whole-part, and tuple-place. We do not rely on a set-theoretic representation of pairs. Rather, we treat pairs as a category of constructed objects, distinct from that of sets. Our generic pattern for constructors relies on plural logic, which does not allow for order. So we reconstruct paired objects by means of special constructed objects marked by positions—we call them "position objects". For pairs, we need two positions, so we introduce two categories of position objects. To emphasize that there is no assumed order, we call these "left objects" and "right objects" (we could have used a different naming convention, such as "red objects" and "blue objects"). These objects are built by the corresponding constructors: the left constructor and the right constructor. For example, suppose that we want to construct the pair of a and b. We first construct from a and b two position objects a' and b', obtained respectively by applying the left constructor to a and the right constructor to b. Then we apply the pair constructor to a' and b', which yields the desired pair.

Second, we do not require that all objects that can be constructed at a stage are constructed at that stage. This enables us to realise separately constructional possibilities that are independent of each other.

Third, the target ontologies involve *traces*, special objects that "log" the structure of the construction process. These characterise how the objects are constructed. For example, suppose a set is constructed at a certain stage. Then

this stage also includes a trace for each element of the set constructed. These traces log the element's view of the construction: the element itself, the type of construction (set), and the output (the set constructed). For example, if the set constructed is $\{a, b\}$, there will be two related traces, one recording that a has been used as input in the construction of a set with output $\{a, b\}$, the other recording that b has been similarly used. More details are provided in Section 9.14.

We describe the process of construction in stage-theoretic terms. The process consists of *stages* ordered by $\trianglelefteq$, which may be seen as an *accessibility relation* in the sense of relational semantics. We think of $s \trianglelefteq t$ quasi-temporally and therefore often pronounce it "s precedes t" or "t is after s". The accessibility relation is reflexive. So the intended sense of "s precedes t" is "s precedes t or s is equal to t".

Every object in the target TLOs is introduced at some stage. We write $x@s$ for "x exists at s". Some objects exist at an initial stage—these are the givens. Once an object exists, it continues to exist at all later stages. However, as we move from earlier to later stages, new objects might be constructed and thus might begin to exist. In the investigation of the target TLOs, this process is called *ontogenesis* as, intuitively, after all the stages all the objects in the ontology exist.

In describing the CCT, we tend to use a dynamic language. This makes the structure of construction easier to visualise, and it exposes relations of ontological dependence among objects (see Section 5). However, the resulting domain is not literally dynamic. It encompasses the objects that exist as a result of the entire process. The same is true of Boolos's stage theory for the iterative conception of set. While his theory describes a seemingly dynamic process of generation, the domain of the theory is not dynamic. It encompasses all sets and results from the entire process.

All construction is effected by means of constructors. A constructor is applied to appropriate objects that exist at some stage so as to construct another object, which may exist only at a later stage. In Appendix B, we provide details about the design choices that inform our approach.

7 Technical background

We aim to make this report, as far as possible, accessible to non-specialist readers, without additional readings. For readers interested in deepening their understanding, appropriate references have been given in the preceding sections. However, the development of the formalisation and the proof of consistency of CCT in the succeeding sections have to rely on specialist and sometimes highly technical notions from logic and mathematics. So these sections of the report are more demanding and require a degree of specialist knowledge.

Section 8 presupposes a basic knowledge of first-order logic. The appropriate background information can be found in logic textbooks such as Enderton 2001 and Hodges 1977. Building on this knowledge, readers should be able to follow

reasonably well the presentation of plural logic in this report. The presentation is self-contained, though interested readers can find a more detailed discussion of plural logic in Florio and Linnebo 2018 and Florio and Linnebo 2021.

The formulation of the CCT in Section 9 is based on, and hence presupposes, the logical framework introduced in Section 8. While no additional knowledge is strictly required to appreciate the formalisation, the context of some parts of the theory (e.g. the axioms of Infinity and Replacement) will be clearer to readers with basic knowledge of set theory (see, for example, Enderton 1977).

The most technical sections of the report—Section 11, Appendix E, and Appendix F—use advanced notions from set theory and model theory. Explanations of the required notions can be found in Kunen 1980 and Chang and Keisler 1990.

To accommodate non-specialist readers, we add informal glosses to the formal statements of the axioms. Moreover, where we make comments, we divide these into more accessible *notes* and technical *remarks*. To further help non-specialist readers, we supply in Appendix I an organised list of background literature.

8 Core Constructional Theory: the formal language

In this section, we introduce the formal language in which the axioms of the CCT will be formulated (for the axioms, see Section 9). The language is given by the logical framework (plural logic) and some non-logical vocabulary. The non-logical vocabulary includes expressions for the key constructional operations and for auxiliary vocabulary needed to express the way in which the constructional operations are deployed through the stages.

8.1 Logical framework

As a logical framework for the CCT, we use plural logic, specifically what is known in the philosophical literature as PFO+, which is short for *plural first-order logic plus plural predicates*. This is the standard language for the formalisation of plural logic, though there are some differences in notation among authors that use this system. Our exposition of the system relies on Florio and Linnebo 2021.

The language of PFO+ extends the language of first-order logic with the following classes of expressions:

A. Plural terms: plural variables ($vv, xx, yy, \ldots$, and variously indexed variants thereof) and plural constants ($aa, bb, \ldots$, and variants thereof), roughly corresponding to the natural language pronoun 'they' and to plural proper names (e.g. 'The Hebrides'), respectively.

B. Universal and existential quantifiers that bind plural variables (e.g. $\forall xx$, $\exists yy$, ...). We may read '$\forall xx$' as 'whenever there are some things xx, then ...', and we may read '$\exists yy$' as 'there are some things yy such that...'.

C. The binary predicate $\prec$ for plural membership, corresponding to the natural language 'is one of' or 'is among'. Its first argument place is singular, while the second is plural (e.g. '$x \prec xx$'). So the predicate expresses a relation between an object and some objects. The intended reading of '$x \prec xx$' is 'x is one of xx', which may be pronounced "x is one of the xs". For example, '$s \prec hh$' may be used to represent 'Skye is one of the Hebrides'.

D. Plural predicates, defined as predicates with one or more argument places reserved for plural terms. An example is $\textsc{Construct}(z, xx, y)$, which represents that z is constructed from xx using the constructor type y. The predicate's arity can be represented by numerical superscripts, which may be omitted for economy, as in the preceding example. The full predicate is $\textsc{Construct}^3(z, xx, y)$.

The non-logical vocabulary of the language, which includes first-order and plural predicates (class D above), is discussed in the next section.

We define a well-formed formula in the language of PFO+ starting with atomic formulas. These are formed by combining predicates with the appropriate number of terms. As noted earlier, we need to ensure that argument places reserved for terms of a certain kind (singular or plural) are occupied by terms of that kind. More complex formulas are obtained by means of sentential connectives and variable binding using singular or plural quantifiers. (See Florio and Linnebo 2021 for details.)

We should emphasize that plural predicates in this language are read collectively. Collective readings of plural predicates contrast with distributive readings. To appreciate the distinction, consider a natural language predicate such as 'lifted a box'. The sentence 'Annie and Bonnie lifted a box' can mean that Annie and Bonnie lifted a box *together* (collective reading), or it can mean that Annie and Bonnie lifted a box *individually* (distributive reading). So a predicate in our formal language such as $\textsc{Construct}(z, xx, y)$ has a collective reading: it means that z is constructed collectively from xx using the constructor type y.

This choice of interpretation for the plural predicates does not make PFO+ any less expressive, since the effect of distributive predication can be obtained by paraphrase. For example, suppose we want to say that xx are sets, which has only a distributive reading, i.e. each of xx is a set. Then, using the singular predicate $\textsc{IsSet}(x)$, we can express that xx are sets by saying that everything that is one of xx is a set:

$$\forall x(x \prec xx \ \rightarrow \ \textsc{IsSet}(x))$$

To illustrate the use of PFO+, it might be helpful to provide some examples of formulas together with informal glosses. For intuitiveness and simplicity, our

informal glosses will often render plural expressions in terms of "pluralities" and their members.

- $v \prec vv$
  (v is one of vv, i.e. it is one of them.)
- $\textsc{AreSeven}(aa)$
  (aa are seven.)
- $\textsc{IsBox}(b) \ \wedge \ \exists xx(\forall x(x \prec xx \ \leftrightarrow$
  $\textsc{IsTile}(x) \ \wedge \ \textsc{IsIn}(x, b)) \ \wedge \ \textsc{Weigh8kg}(xx))$
  (The tiles in box b weigh 8 kg together. Literally, b is a box and there are some things such that anything is one of them if and only if it is a tile in b and, together, they weigh 8 kg.)
- $\exists z \exists xx \exists y \textsc{Construct}(z, xx, y)$
  (Some object is constructed from some things using some type of constructor.)
- $\exists zz \forall z(z \prec zz \leftrightarrow (z = a \vee z = b))$
  (Consider two objects a and b, possibly identical. There is a plurality that has a and b, and only a and b, as its members. If $a = b$, then this is a "singleton" plurality.)
- $\forall z \forall xx \forall y(\textsc{Construct}(z, xx, y) \rightarrow \forall ww(\forall w(w \prec xx \leftrightarrow w \prec ww) \rightarrow$
  $\textsc{Construct}(z, ww, y)))$
  (If an object of some type is constructed from a plurality xx, then it is also constructed by any plurality ww that has exactly the same members as xx.)

8.2 Non-logical vocabulary

We have designed the language of the CCT so that its basic non-logical vocabulary can express the constructional operations in scope, the structure of the stages, and the way in which operations and stages interact to produce the target ontology. This vocabulary is given by constants, first-order predicates, and plural predicates of kind D mentioned above.

The present version of the CCT includes a single, generic constructor, which we express here as a three-place predicate rather than an operator. Given our requirements for formalisation, this is the simpler choice: it makes working in classical logic unproblematic, as we avoid the need to handle empty terms and partial functions. Different types of constructed object can be obtained by switching one of its parameters. The switch is effected using special constants, one for each type of constructed objects.

Generic constructor	
predicate	*intended reading*
$\textsc{Construct}(z, xx, y)$	z is an/the object constructed from xx using the type y

The predicate has three argument places. The first is associated to the object constructed, while the other two are associated to the inputs of the construction (a plurality of objects) and an object representing the type of construction effected.

To indicate the types of construction, we use six special constants.

Types	
constant	*intended reading*
c_{set}	set construction type
c_{sum}	sum construction type
c_{left}	left position construction type
c_{right}	right position construction type
c_{pair}	pair construction type
c_{union}	union construction type

For example, we describe the construction of a set by filling the third argument place of $\textsc{Construct}(z : xx, y)$ with the constant for the desired type of construction: $\textsc{Construct}(z : xx, c_{\text{set}})$.

Next, we have some predicates pertaining to stages and their structure.

Stages	
predicate	*intended reading*
$\textsc{Stage}(x)$	x is a stage
$\trianglelefteq(s, t)$	s is before, or equal to, t
$@(x, s)$	x exists at (stage) s

The process of construction involves the application of the generic constructor to some objects and the specification of the type of construction to be effected. Since we require pluralities to be non-empty (see Section 9), we thereby

disallow empty inputs to any constructor. Moreover, one can refine this process by placing constraints—in accordance with the target TLOs—upon what objects can serve as inputs relative to particular types. In the case of the set constructor, for example, all objects other than position objects can serve as inputs and thus are (informally) "settable". In the case of mereological sums, the CCT restricts the process of construction to *individuals*, namely, objects that are either givens or sums. Only individuals are "summable". The notion of "individual" is expressed by a primitive predicate.

Individuals	
predicate	*intended reading*
INDIVIDUAL(x)	x is either a given or a sum

As mentioned above, the target ontology involves traces, namely, objects that log the structure of the construction process. The components of this structure are accessed by three predicates listed below. More details about these predicates and their role is provided in Section 9.14.

Traces	
predicate	*intended reading*
HASOUTPUT(w, z)	w records a construction with output z
HASINPUT(w, x)	w records a construction with x as one of the inputs
HASTYPE(w, y)	w records a construction of type y

This completes our basic stock of primitive non-logical vocabulary. We will make frequent use of definitions to introduce new notions. To facilitate the automated translation of the formalisation into computer-readable format, we treat these definitions as axioms rather than abbreviations. As a result, these definitions introduce new primitive non-logical expressions relying on non-logical expressions from the basic stock as well as prior definitions.

8.3 Conventions

To simplify the exposition of the CCT, we rely on a number of familiar conventions. Let us highlight some key examples.

First, we adopt the following formatting conventions.

- When parentheses are omitted, ambiguities should be resolved by applying the standard rules of precedence among logical connectives.

- Different styles of parentheses are used to make groupings of formulas easier to recognize.
- A leading block of universal quantifiers is frequently omitted. So axioms with free variables, i.e. variables not bound by a quantifier, are short for their universal closure.
- To improve readability, we use infix notation for some predicates. For example, we write '$s \trianglelefteq t$' rather than '$\trianglelefteq(s,t)$', and '$x@t$' rather than '$@(x,t)$.

Secondly, we adopt a convention related to restricted quantification.

- Quantification restricted to certain kinds of entities may be marked by the use of particular variables. We use s, s', s_0, t, etc. to range over stages. So $\forall s\varphi(s)$ is short for $\forall s(\textsc{Stage}(s) \rightarrow \varphi(s))$. Similarly, we use ss, ss', tt for plural quantification over stages. So $\forall ss\varphi(ss)$ is short for $\forall ss(\forall x(x \prec ss \rightarrow \textsc{Stage}(x)) \rightarrow \varphi(ss))$.

Finally, we use the following, less common formatting convention in order to emphasize constructional links among argument places of relevant predicates.

- We use a colon rather than a comma to separate the argument place for the constructed object from the rest of the arguments. For example, we write $\textsc{Construct}(z : xx, y)$ rather than $\textsc{Construct}(z, xx, y)$ to indicate that z is constructed from xx and y.

9 Core Constructional Theory: the axioms

Our theory has several parts, with different concerns. The axiomatisation aims to be perspicuous, user-friendly, and modular rather than minimal. It is based on the language introduced in the preceding section.

9.1 Plural logic

The formal system PFO+, which we introduced in Section 8.1, comes equipped with logical axioms and rules of inference. The axioms and rules associated with the logical vocabulary of ordinary first-order logic are the standard ones. For example, one could rely on introduction and elimination rules for each logical expression. The plural quantifiers are governed by axioms or rules analogous to those governing the first-order quantifiers.

In addition, we have the following axioms or axiom schemes. First, every plurality is non-empty:

(A1) $\forall xx \exists y\ y \prec xx$

(Every plurality has at least one member.)

Then, there is an axiom scheme of indiscernibility:

(A2) $\forall xx \forall yy(\forall z(z \prec xx \leftrightarrow z \prec yy) \rightarrow (\varphi(xx) \leftrightarrow \varphi(yy)))$

(Coextensive pluralities satisfy the same formulas.)

Some remarks are in order. First, the formula φ may contain parameters. So we have the universal closure of each instance of the displayed axiom scheme. Second, as is customary, we write $\varphi(xx)$ for the result of replacing all free occurrences of some designated plural variable vv with 'xx' whenever 'xx' is substitutable for vv in φ (see, for example, Enderton 1977, p. 113). Third, (A2) is a plural analogue of Leibniz's law of the indiscernibility of identicals, and as such, the scheme needs to be restricted to formulas $\varphi(xx)$ that don't set up intensional contexts. Since there is no requirement for intensional contexts in the CCO, no such context will be introduced in the CCT. Thus, the axiom scheme may be used unrestrictedly in the present setting.

Finally, there is the unrestricted axiom scheme of *plural comprehension*, an intuitive principle that provides information about what pluralities there are. For any formula $\varphi(x)$ in which x is free but xx is not, we have an axiom stating that if $\varphi(x)$ is satisfied by at least one thing, then there are the things each of which satisfies $\varphi(x)$:

(A3) $\exists x\varphi(x) \rightarrow \exists xx \forall x(x \prec xx \leftrightarrow \varphi(x))$

(If something is φ, then there are some things that are all and only the φs. That is, if something is φ, then the φs exist.)

We refer to an axiomatization of plural logic based on the principles just described as *full plural logic*. The fullness of the logic has to do with the fact that plural comprehension applies to any formula $\varphi(x)$—provided, as stated, that x but not xx occur free.

Two important relations in plural logic are introduced as definitions. One is the many-many relation of plural inclusion ('are among'), which is symbolized as '$\preccurlyeq$':

(D1) $xx \preccurlyeq yy \leftrightarrow \forall z(z \prec xx \rightarrow z \prec yy)$

(Some things xx are among yy when everything that is one of xx is one of yy.)

Another important relation is plural identity, symbolized as '$\approx$'. The relation can be defined using '$\preccurlyeq$', the symbol introduced by the preceding definition:

(D2) $xx \approx yy \leftrightarrow (xx \preccurlyeq yy \wedge yy \preccurlyeq xx)$

(Two pluralities xx and yy are identical if and only if xx are among yy and yy are among xx, i.e. xx and yy have the same members.)

When showing how to derive the axioms of set theory from the CCT (Section 10.2), we will mention another principle of plural logic, one that corresponds to the Axiom of Choice in set theory. Informally, the plural principle states that

for every plurality *pp* representing a family of pairwise disjoint pluralities of objects, there is a *choice plurality* for *pp*. This is a plurality containing a unique member for any plurality in the family represented by *pp*. (See Section 10.2 for more details.) We do not adopt this plural principle as an official axiom of CCT. Rather, we leave it as an option available in contexts where the set-theoretic Axiom of Choice is needed.

9.2 Stages

The stages are linearly ordered by the accessibility relation $\trianglelefteq$:

(A4) $\forall s\ s \trianglelefteq s$

($\trianglelefteq$ is reflexive.)

(A5) $\forall s \forall t (s \trianglelefteq t\ \wedge\ t \trianglelefteq s\ \rightarrow s = t)$

($\trianglelefteq$ is antisymmetric.)

(A6) $\forall s_0 \forall s_1 \forall s_2 (s_0 \trianglelefteq s_1\ \wedge\ s_1 \trianglelefteq s_2\ \rightarrow s_0 \trianglelefteq s_2)$

($\trianglelefteq$ is transitive.)

(A7) $\forall s \forall t (s \trianglelefteq t\ \vee\ t \trianglelefteq s)$

($\trianglelefteq$ is connected.)

In addition, stages are well-founded by $\trianglelefteq$. Let us define $s \triangleleft t$ ("s is strictly before t") as follows:

(D3) $s \triangleleft t \leftrightarrow (s \trianglelefteq t \wedge s \neq t)$

Then, we have:

(A8) $\forall ss \exists s (s \prec ss \wedge \neg \exists t (t \prec ss \wedge t \triangleleft s))$

(The stages are well-founded by $\trianglelefteq$. That is, for any plurality ss of stages, there is a member s that is first among ss with respect to the order $\trianglelefteq$.)

Notice that the last axiom makes essential use of plural logic, asserting something of every plurality of stages. Moreover, this axiom enables us to do well-founded induction on $\trianglelefteq$. (Assume that a property holds at the initial stage and that, whenever the property holds at every s such that $s \triangleleft t$, then it also holds at t. Then the property holds at every stage.) Using this form of induction, we will be able to show, for example, that something exists at every stage.

The accessibility relation is also serial:

(A9) $\forall s \exists t\ s \triangleleft t$

(For every stage, there is a strictly later stage.)

We say that t *is a successor stage of* s if and only if s precedes t and there is no stage strictly in between s and t, that is:

(D4) $\textsc{Succ}(s,t) \leftrightarrow s \lhd t \wedge \neg\exists u\ (s \lhd u \wedge u \lhd t)$

Later we will introduce a related abbreviation, a predicate '$\textsc{Max}(s,t)$' representing that t is a maximal extension of s, that is, that t contains the result of carrying out every construction that is possible at s (Section 9.15).

Remark. Axiom (A9) and the well-foundedness of $\trianglelefteq$ entail that every stage has a successor stage.

Remark. We will eventually see that the seriality of $\trianglelefteq$ can be proved from the other axioms of our theory (see Section 9.9). We keep the axiom of seriality, however, as we will sometimes be interested in working with less than the full theory.

There is an infinite stage:

(A10) $\exists t(\exists s\, s \lhd t \wedge \forall s(s \lhd t \rightarrow \exists u(s \lhd u \wedge u \lhd t)))$

(There is a stage that is after some stage and is not immediately after any other stage.)

Note. To see how this axiom secures an infinite stage, consider the finite stages $0, 1, 2, ...$ and the first infinite stage ω, which comes after all the finite stages. To say that a stage t is not immediately after any other stage means that, if there is a stage s before t, then there is a stage between s and t. Both 0 and ω are not immediately after any other stage. (In the case of 0, this condition is vacuously satisfied). However, unlike 0, ω is after some stage. So we can characterise ω as the first stage that is after some stage but is not immediately after any other stage.

Remark. We can now prove that there is an initial stage. Axiom (A10) entails that there is a stage. Thus the plurality of stages exists and, given (A8), is well-founded. So there is a first stage.

Let us say that some things exist at a stage if and only if each of them exists at that stage. So we define '$xx@@s$' ("xx exist at s") as follows:

(D5) $xx@@s \ \leftrightarrow\ \forall x(x \prec xx \rightarrow x@s)$

We also assume a stage-theoretic version of the axiom scheme of Replacement. This is a powerful principle, which provides information about how many stages there are. Specifically, the principle says there are so many stages that the objects available at any one of them do not suffice to reach arbitrarily high in the hierarchy of stages. That is, for any stage s, any objects xx at s, and any formula $\psi(x,y)$ that represents a function from xx to objects throughout the hierarchy, there is a stage at which we find all the values that this function takes for arguments among xx.

(A11)

$$xx@@s \wedge \forall x(x \prec xx \rightarrow \exists y(\neg \textsc{Stage}(y) \wedge \forall z(\psi(x,z) \leftrightarrow y = z))) \;\rightarrow \\ \exists t \forall x(x \prec xx \rightarrow \forall y(\psi(x,y) \rightarrow y@t))$$

(Suppose that xx exist at s and that $\psi(x,y)$ represents a function mapping objects among xx to objects other than stages. Then there is a stage t such that what exists at t includes the image under ψ of every member of xx.)

This axiom is included primarily for the sake of re-construing set theory; it is not essential to our approach. Notice, though, that our Replacement axiom is not about sets but about stages and the many objects that exist at various stages. Indeed, Replacement is the first of our axioms that ties the existence of stages to the existence of objects that exist at stages, i.e. it is the first "interactive" axiom.

9.3 Initial stage

An initial stage is one that is not preceded by any other stage. Let us define '$\textsc{Init}(s)$' to mean that s is initial:

(D6) $\textsc{Init}(s) \leftrightarrow \neg\exists t\, t \lhd s$

Remark. It follows from the connectedness of $\trianglelefteq$ that s is initial if and only if it is before every other stage, i.e. $\forall t\, s \trianglelefteq t$. This also means that there is at most one initial stage. In a different context, the current definition of initial stage would permit several alternative initial stages.

We define an object to be a given if and only if it exists at some initial stage:

(D7) $\textsc{Given}(x) \leftrightarrow \exists s(\textsc{Init}(s) \wedge x@s)$

Then we assume the existence of some given:

(A12) $\exists x\, \textsc{Given}(x)$

(There is some given.)

In our setting, we need objects that code information about types of construction. So we assume the existence of such objects. For example, we introduce an object c_{set} that stands for the type *set*. We do the same for each type of construction in scope: *sum*, *left object*, *right object*, *pair*, and *union*. We make the pragmatic choice to treat these objects as givens and, in fact, as the only givens. In the future, we might want to separate these objects from those that more properly represent the intended constructional ontology. For this specific implementation of the CCT, and to allow modularity, we have the following strengthening of (A12):

(A13) $\textsc{Given}(c_{\text{set}}) \ \wedge \ \textsc{Given}(c_{\text{sum}}) \ \wedge \ \textsc{Given}(c_{\text{left}}) \ \wedge \ \textsc{Given}(c_{\text{right}}) \ \wedge \ \textsc{Given}(c_{\text{pair}}) \ \wedge \ \textsc{Given}(c_{\text{union}})$

(The givens include: c_{set}, c_{sum}, c_{left} c_{right}, c_{pair}, c_{union}.)

These are six distinct givens:

(A14) $c_{\text{set}} \neq c_{\text{sum}} \ \wedge \ c_{\text{set}} \neq c_{\text{left}} \ \wedge \ c_{\text{set}} \neq c_{\text{right}} \ \wedge \ c_{\text{set}} \neq c_{\text{pair}} \ \wedge \ c_{\text{set}} \neq c_{\text{union}} \ \wedge \ c_{\text{sum}} \neq c_{\text{left}} \ \wedge \ c_{\text{sum}} \neq c_{\text{right}} \ \wedge \ c_{\text{sum}} \neq c_{\text{pair}} \ \wedge \ c_{\text{sum}} \neq c_{\text{union}} \ \wedge \ c_{\text{left}} \neq c_{\text{right}} \ \wedge \ c_{\text{left}} \neq c_{\text{pair}} \ \wedge \ c_{\text{left}} \neq c_{\text{union}} \ \wedge \ c_{\text{right}} \neq c_{\text{pair}} \ \wedge \ c_{\text{right}} \neq c_{\text{union}} \ \wedge \ c_{\text{pair}} \neq c_{\text{union}}$

It is convenient to have a separate predicate applying to these objects *qua* representations of types of construction:

(D8)
$$\textsc{Type}(y) \ \leftrightarrow (y = c_{\text{set}} \ \vee \ y = c_{\text{sum}} \ \vee \ y = c_{\text{left}} \ \vee \ y = c_{\text{right}} \ \vee \ y = c_{\text{pair}} \ \vee \ y = c_{\text{union}})$$

In section 9.13, we will see that the type union has a *derived* status, while the remaining types are *basic*.

Remark. Our approach can be extended to other constructors one may wish to add. If new constructors are adopted, then the definition of $\textsc{Type}(y)$ will have to be modified accordingly. However, as we will see, other axioms that utilise this predicate can remain unchanged, thus allowing a pleasing modularity.

9.4 What exists at stages

Some of the previous axioms already provide information about what exists at stages. For example, stipulating the existence of some givens implies that they exist at the initial stage. Moreover, the axiom scheme of Replacement informs us that if some objects exist at a stage, their images under a functional relation also exist at some stage—provided only that no image is a stage. It is worth emphasising that while stages are part of the CCT, they are not part of the CCO. We use the term 'object' in two corresponding ways. In a wide sense, it stands for all entities in the domain of the CCT. In a narrow sense, it stands for relevant entities in the CCO and thus excludes stages. (So far we have used the term primarily in the narrow sense.) We now continue to explore principles that connect stages and what exists.

First, we have an axiom that ties the objects in our ontology to stages, with the exception of the stages themselves, which are not part of the intended ontology and thus have a purely auxiliary role. (In future work, we will explore formalisations that do rely on stages as primitive.)

(A15) $\forall x(\neg\textsc{Stage}(x) \rightarrow \exists s \, x@s)$

(Everything that isn't a stage exists at some stage.)

Note. We will be able to prove the other direction of the conditional in (A15) relying also on axioms introduced in later sections. So, in the CCT, something is not a stage if and only if it exists at some stage.

We identify stages extensionally. That is, stages with identical domains are identical:

(A16) $\forall x(x@s \leftrightarrow x@t) \rightarrow s = t$

(Stages with identical domains are identical.)

Moreover, stages are "cumulative":

(A10) $s \trianglelefteq t \wedge x@s \rightarrow x@t$

(When one stage precedes another, then everything that exists at the former also exists at the latter.)

Define a stage t to be the least upper bound (LUB) of some stages ss in the usual way:

(D9) $\mathrm{LUB}(t, ss) \leftrightarrow \forall s(s \prec ss \rightarrow s \trianglelefteq t) \wedge \forall t'(\forall s(s \prec ss \rightarrow s \trianglelefteq t') \rightarrow t \trianglelefteq t')$

(According to the definition, t is the least upper bound of ss if and only if two conditions are met: (i) t is an upper bound of ss, i.e. t is after, or equal to, any stage in ss; (ii) t is the least among the upper bounds of ss, i.e. t is before, or equal to, any upper bound of ss.)

Next, we have an axiom characterising "collection" stages. No new object is introduced at these stages; they merely collect objects existing at previous stages. Collection stages are useful when, as in our case, the construction process is extended into the transfinite and hence requires infinite stages. To see why, consider the first infinite stage ω, which comes after all finite stages. We cannot think of ω as introducing new objects constructed from those available at the preceding stage. This is because there is no stage that immediately precedes ω: any stage n prior to ω has a successor stage $n + 1$ that is also prior to ω. So what happens at stage ω? We adopt the option of simply letting ω collect the objects that exist at prior stages. The same applies to other infinite stages that are limit, namely, neither an initial stage nor a successor one. The following entails that limit stages are collection stages (and it provides trivial information about successor stages):

(A17) $\mathrm{LUB}(t, ss) \rightarrow \forall x(x@t \rightarrow \exists s(s \prec ss \wedge x@s))$

(Suppose t is the least upper bound of some stages ss. Then everything that exists at t exists at some of ss.)

Remark. The cumulativity of the stages entails the other direction, which means this conditional can be strengthened to a biconditional.

We now want to define a predicate $\textsc{ConstrFrom}(x, s)$ whose intuitive meaning is that x is constructed from some objects that exist at stage s. The precise definition is as follows:

(D10) $\textsc{ConstrFrom}(x, s) \leftrightarrow \exists xx \exists y(xx@@s \wedge \textsc{Type}(y) \wedge \textsc{Construct}(x : xx, y))$

Finally, we restrict what can exist at a successor stage. This axiom also uses the notion of trace, which is characterised in Section 9.14.

(A18) $\textsc{Succ}(s, t) \wedge x@t \ \rightarrow \ x@s \vee \textsc{ConstrFrom}(x, s) \vee \textsc{Trace}(x)$

(Everything that exists at a successor stage either existed at the predecessor stage, or is constructed from something at that stage, or is a trace.)

Subsequent axioms will ensure the existence of appropriate traces at each successor stage. In particular, they will ensure that new traces emerging at a successor stage correspond to constructions effected at that stage.

9.5 What is and is not constructible

We impose various constraints on which objects can serve as inputs to different types of construction. These constraints do not appear in the framework of Fine 2010. They are added here to reflect the constraints in the target TLOs and could easily be lifted in connection with the pursuit of other goals. In this section, we provide an informal characterisation of the constraints. Formal renderings of them will be developed in the later sections, where each type of construction is axiomatised.

Let us recapitulate the overall structure of our ontology. We have five basic types of objects that can be constructed: set, sum, two types of position objects (left object and right object), and pair. One additional type of objects is derived: union. The construction process proceeds in stages, beginning with some givens. It yields objects of basic and derived types as well as traces storing information about the constructions effected.

The construction of sets is the least constrained operation. Sets, sums, pairs, givens, and traces can all serve as inputs to constructions of sets or, as we may put it, are "settable". Moreover, any plurality mixing objects of settable types is itself settable. By contrast, the construction of sums is the most constrained operation. Only individuals can serve as inputs, where an individual is defined as follows:

(D11) $\textsc{Individual}(x) \leftrightarrow (\textsc{Given}(x) \ \vee \ \exists xx \ \textsc{Construct}(x : xx, c_{\text{sum}}))$

(Any object is an individual if and only if it is either a given or a sum.)

Remark. The axioms of the CCT entail that sets, position objects, and pairs are not individuals. More generally, any two objects constructed from distinct types other than union are distinct. Unions are sets. (See Section 9.8.)

Position objects resemble sets in the following respect: the objects that can be used as inputs for their construction are exactly sets, individuals (i.e. givens and sums), pairs, and traces. So position objects are not themselves "positionable". There is a difference, however. The input to the construction

of a position object must be a "singleton plurality", namely, a plurality with only one member. So any set, sum, pair, or trace can be the single member of a plurality eligible for the construction of a left or right object.

Pairs can be constructed from position objects and only from them. But the configuration of the input plurality must respect two conditions: (i) the plurality contains exactly two members; (ii) one member is a left object while the other member is a right object.

To sum up, the constraints vary according to the type of construction. The difference might concern the type of objects in the input plurality or the size of that input.

9.6 Generic constructor

The generic constructor is functional in the following sense:

(A19)
$$\textsc{Construct}(z_1 : xx_1, y_1) \ \wedge \ \textsc{Construct}(z_2 : xx_2, y_2) \wedge$$
$$y_1 = y_2 \ \wedge \ xx_1 \approx xx_2 \ \rightarrow \ z_1 = z_2$$

(Any two objects of the same type, constructed from the same pluralities, are the same.)

We also require that the third argument place be a type.

(A20) $\textsc{Construct}(z : xx, y) \ \rightarrow \ \textsc{Type}(y)$

(If an object z is constructed from xx using y, then y is a type.)

9.7 Specialised constructors

We need axioms that characterise the behaviour of the four basic types of construction and fix the identity criteria of their outputs. This is done in the next four sections, which cover respectively sets, sums, position objects, and pairs.

For ease of exposition, it is useful to define operations that represent the construction of objects of each type.

(D12) $\textsc{Set}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{set}})$

(D13) $\textsc{Sum}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{sum}})$

(D14) $\textsc{Left}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{left}})$

(D15) $\textsc{Right}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{right}})$

(D16) $\textsc{Pair}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{pair}})$

(D17) $\textsc{Union}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{union}})$

It is then convenient to speak as if we are dealing with different types of constructor. For example, we will refer to the "set constructor" and describe its properties. However, it is important to keep in mind that the CCT is built around a single, generic constructor that can be specialised to obtain different types of objects. So the notion of "set constructor" is ultimately understood in terms of applications of the generic constructor using the parameter c_{set}.

We use the following definitions to capture the extension of each type of constructed object.

(D18) $\textsc{IsSet}(x) \leftrightarrow \exists xx\textsc{Set}(x : xx)$

(D19) $\textsc{IsSum}(x) \leftrightarrow \exists xx\textsc{Sum}(x : xx)$

(D20) $\textsc{IsLeft}(x) \leftrightarrow \exists xx\textsc{Left}(x : xx)$

(D21) $\textsc{IsRight}(x) \leftrightarrow \exists xx\textsc{Right}(x : xx)$

(D22) $\textsc{IsPair}(x) \leftrightarrow \exists xx\textsc{Pair}(x : xx)$

(D23) $\textsc{IsUnion}(x) \leftrightarrow \exists xx\textsc{Union}(x : xx)$

We wish to highlight an important feature possessed by some but not all types of constructed objects. Unlike sums, sets, positions objects, and pairs are strictly dependent on their members: if two objects of the relevant type are identical, they are constructed from the same plurality. In other words, an object of the relevant type is constructed from at most one plurality. This feature corresponds to the *injectivity* of the relevant type, which we express as follows.

(D24)

$$\begin{aligned}\textsc{Injective}(y) \;\leftrightarrow\; &\textsc{Type}(y) \wedge \\ &\forall z_1 \forall xx_1 \forall z_2 \forall xx_2 (\textsc{Construct}(z_1 : xx_1, y) \;\wedge\; \textsc{Construct}(z_2 : xx_2, y) \;\rightarrow \\ &\qquad (z_1 = z_2 \;\rightarrow\; xx_1 \approx xx_2))\end{aligned}$$

9.8 Classification

We want to classify the objects in the domain of the CCT. Our target principles are:

(i) everything is either an individual, a set, a left object, a right object, a pair, a trace, or a stage;

(ii) nothing has more than one of the mentioned properties.

The first principle can be obtained from the preceding axioms. This can be seen as follows. By the axioms concerning what exists at a stage, we have that anything that is not a stage exists at some stage. The only things that exist at stages are givens, constructed objects, and traces. Givens are individuals, and

a constructed object is a set, a sum (thus an individual), a position object, or a pair. A union is a set and thus does not introduce an extra dimension in the classification.

The second principle is assumed as an axiom.

(A21)

$$\forall z_1 \forall z_2 \forall xx_1 \forall xx_2 \forall y_1 \forall y_2 (\textsc{Construct}(z_1 : xx_1, y_1) \wedge \textsc{Construct}(z_2 : xx_2, y_2) \\ \wedge y_1 \neq y_2 \wedge y_1 \neq c_{\text{union}} \wedge y_2 \neq c_{\text{union}} \rightarrow z_1 \neq z_2) \wedge \\ \forall z(\exists xx \exists y \textsc{Construct}(z : xx, y) \rightarrow \neg \textsc{Trace}(z) \wedge \neg \textsc{Stage}(z)) \wedge \\ \forall z(\textsc{Trace}(z) \rightarrow \neg \textsc{Stage}(z))$$

(Nothing has more than one of the relevant properties. More explicitly, any two objects constructed from distinct types other than union are distinct; constructed objects are distinct from traces and stages; and traces are distinct from stages.)

Note. The type of the union constructor must be excluded since unions are sets. However, the characterisation of unions (Section 9.13) ensures that they identified with sets.

9.9 Set constructor

We begin by defining the permissible inputs to the set constructor, as anticipated in Section 9.5:

(D25)

$$\textsc{Settable}(x) \leftrightarrow \\ (\textsc{IsSet}(x) \vee \textsc{Individual}(x) \vee \textsc{IsPair}(x) \vee \textsc{Trace}(x))$$

We use the definition to constrain the application of the set constructor:

(A22) $\forall xx \forall x(\textsc{Set}(x : xx) \rightarrow \forall z(z \prec xx \rightarrow \textsc{Settable}(z)))$

(If some things construct a set, then each of them is settable.)

The next axioms specify the relation between sets, their members, and stages. We require that every appropriate plurality existing at a stage is eventually used to construct a set:

(A23)

$$\forall xx \forall s(xx@@s \wedge \forall x(x \prec xx \rightarrow \textsc{Settable}(x)) \rightarrow \\ \exists x \textsc{Set}(x : xx))$$

(For every plurality xx of settable objects existing at s, the set of xx exists.)

Analogous requirements will apply to other constructors. We also lay down that sets strictly depend on their elements:

(A24) $\textsc{Injective}(c_{\text{set}})$

(The type set is injective. That is, a set is constructed from at most one plurality.)

Remark. The axioms of the CCT entail that the elements of a set exist at an earlier stage than the set itself:

$$\forall x \forall xx (x@t \wedge \textsc{Set}(x : xx) \rightarrow \exists s (s \lhd t \wedge xx@@s))$$

We reason as follows. Suppose that a set x exists at a stage. Because limit stages are collection stages, x must exist at the initial stage or at a successor stage. It cannot exist at the initial stage, since the givens are not sets. So x must exist at a successor stage. Let t be the least successor stage at which x exists. Since x does not exist prior to t and it is not a a trace, (A18) yields that the elements of x exist at an earlier stage than x. This reasoning is based on previous axioms, including the fact that the basic kinds of objects (sets, individuals, position objects, pairs, and traces) do not overlap, a fact sanctioned by the classification axiom (A21).

In this setting, there is a natural definition of set-theoretic membership:

(D26) $x \in y \;\leftrightarrow\; \exists yy (\textsc{Set}(y : yy) \wedge x \prec yy)$

That is, to be an element of a set is to be a member of the (unique) input plurality from which the set is constructed.

Remark. Every set is constructed from some plurality and thus has the members of that plurality as elements. Since pluralities are not empty, it follows that there is no empty set in this setting. Moreover, the functionality of the generic constructor and the injectivity of the set constructor yield an analogue of the usual extensionality criterion for set: two sets are identical if and only if they are constructed from the same pluarlity. In symbols:

$$\textsc{Set}(x : xx) \wedge \textsc{Set}(y : yy) \rightarrow (xx \approx yy \leftrightarrow x = y)$$

Remark. In Appendix D, we present and discuss four key principles for the identity of constructed entities (Fine 2010): Collapse (C), Leveling (L), Absorption (A), and Permutation (P). Here we simply note that the preceding axioms enable us to derive the correct CLAP profile for sets, namely, $\not{C}\not{L}$AP.

Remark. Given the set constructor and the axioms governing it, it follows that the stages are serial, that is, for every stage, there is a strictly later stage: $\forall s \exists t\, s \lhd t$. The argument exploits a version of Russell's paradox. Consider a stage s. Use plural comprehension to consider the plurality xx of all and only those objects at s that are not elements of themselves. This plurality can be

used to construct a set y at a stage t, with $s \trianglelefteq t$. We want to show that t is strictly after s: $s \triangleleft t$. Suppose not, i.e. that $t = s$. We now ask whether $y \in y$, with $\in$ defined by (D26). Since y has as its elements all and only the objects at s that are not elements of themselves, and since y by assumption exists at s, it follows that y is an element of itself if and only if it is not an element of itself. Since this is a contradiction, we conclude that $s \neq t$ and thus that $s \triangleleft t$ after all, as desired.

In Section 10.1 we state the axioms of Zermelo-Fraenkel set theory with the Axiom of Choice, modified to fit the present context. We then show how to derive such axioms from those of the CCT (Section 10.2). To facilitate the derivation, we adopt an alternative, set-theoretic version of the axiom that there is an infinite stage (Section 9.2):

(A25)

$$\exists xx \exists s \exists y(xx@@s \ \wedge \ \textsc{Given}(y) \ \wedge \ y \prec xx \ \wedge \ \forall x(x \prec xx \ \rightarrow \\ \exists u \exists uu(\textsc{Set}(u : uu) \ \wedge \ \forall w(w \prec uu \ \leftrightarrow \ w = x) \ \wedge \ u \prec xx)))$$

(There are some xx such that xx exist at a stage, xx have a given as member and, whenever x is a member of xx, its singleton $\{x\}$ is also a member of xx.)

Once this axiom is adopted, we could dispense with the previous assumption that there is an infinite stage. For the plurality required to exist here can only arise at a limit stage.

9.10 Sum constructor

The inputs of the sum constructor are constrained to include only individuals:

(D27) $\textsc{Summable}(x) \ \leftrightarrow \ \textsc{Individual}(x)$

We make the corresponding requirement:

(A26) $\forall xx \forall x(\textsc{Sum}(x : xx) \ \rightarrow \ \forall z(z \prec xx \ \rightarrow \textsc{Summable}(z)))$

(If some things construct a sum, then each of them is summable.)

Remark. It follows from our assumptions that all givens are summable.

As with sets and other types of constructed objects, we require that every appropriate plurality existing at a stage is eventually used to construct a sum:

(A27) $\forall z(z \prec xx \rightarrow \textsc{Summable}(z) \wedge z@s) \rightarrow \exists x \textsc{Sum}(x : xx)$

(For every plurality xx of summable objects existing at a stage s, the sum of xx exists.)

Note. In the case of sets, we have that the elements of any set exist at a stage strictly before any stage at which the set itself exists. Does something analogous hold of sums?

The answer is negative: sums are simply different from sets in this regard. To see this, consider a given g and let gg be the plurality whose sole member is g. Then, as we will see shortly, $\textsc{Sum}(g : gg)$. This provides a counterexample to the hypothesis that every sum is the sum of some objects that exist strictly before the sum itself.

Next, we need to state a criterion of identity for sums. Before we do that, however, we state two constraints on their individuation, which Fine (2010) labels *Collapse* and *Leveling* (see Appendix D for more context). First, sums satisfy Collapse:

(A28) $\textsc{Sum}(x : xx) \wedge \forall u(u \prec xx \leftrightarrow u = y) \rightarrow x = y$

(The sum constructed from the singleton plurality of x is x.)

Second, sums satisfy Leveling:

(A29)

$$\begin{aligned}
&\textsc{Sum}(x : xx) \ \wedge \ \textsc{Sum}(y : yy) \ \wedge \\
&\quad \forall z_1(z_1 \prec xx \ \rightarrow \ (z_1 \prec yy \ \vee \exists zz(zz \preccurlyeq yy \ \wedge \ \textsc{Sum}(z_1 : zz)))) \ \wedge \\
&\quad \forall z_2(z_2 \prec yy \ \rightarrow \ (z_2 \prec xx \ \vee \\
&\qquad \exists zz(zz \preccurlyeq yy \ \wedge \ z_2 \prec zz \ \wedge \ \exists z_3(z_3 \prec xx \ \wedge \ \textsc{Sum}(z_3 : zz))))) \ \rightarrow \\
&\qquad\quad x = y
\end{aligned}$$

(Suppose x is a sum of xx and y is a sum of yy. If the following two conditions are satisfied, then x is y:

1. every x among xx is either one of yy or is a sum of some zz among yy;
2. every y among yy is either one of xx or is one of some members of yy whose sum is among xx.

In other words, two pluralities, one obtained from the other by replacing some pluralities of objects with their respective sums, yield the same sum.)

Remark. Given the axioms of plural logic, the functionality of the generic constructor, (A19), entails Absorption and Permutation for any type of constructor. Thus, (A19), (A28), and (A29) lay down the entire CLAP profile of sums, namely, full CLAP (see Appendix D).

Axioms (A19), (A28), and (A29) provide *sufficient* conditions for two mereological sums to be identical. But we do not yet have any *necessary* conditions. Let us explore the possibility of adding an "extremal clause" to the effect that

no two sums are identical unless they are identified as a result of these three axioms. The three axioms can then be regarded as a recursive definition of sum identity. Our question, then, is how this extremal clause might be formulated.

We first observe that we can prove that *every sum is a sum of givens*. The proof goes by induction on the construction of sum (see Section 9.16 for a general version of induction on construction). Let us sketch the reasoning, which relies essentially on the fact that givens and sums are individuals and that only individuals are summable. First, by Collapse, every given is a sum of givens. Second, assuming that each of some summable objects xx is a sum of givens, it follows by Leveling that so too is the sum of xx. Thus, the promised conclusion follows by induction on stages.

This observation means that, to provide a criterion of identity for sums, it suffices to state a criterion for sums of givens. In fact, it turns out there is a simple way to do so, that is, to adopt the the "hyperextensional" principle of Goodman 1958:

(A30)

$$\forall x_1(x_1 \prec xx_1 \rightarrow \textsc{Given}(x_1)) \ \wedge \ \forall x_2(x_2 \prec xx_2 \rightarrow \textsc{Given}(x_2)) \wedge \textsc{Sum}(z_2 : xx_1) \ \wedge \ \textsc{Sum}(z_2 : xx_2) \rightarrow (z_1 = z_2 \ \rightarrow \ xx_1 \approx xx_2)$$

(If two sums of givens are identical, those givens must also be identical. That is, a sum can be constructed from givens xx_1 and also from givens xx_2, then xx_1 are the very same objects as xx_2.)

This is pleasingly analogous to the strict dependence that holds for sets, captured by the injectivity of the set constructor. Since every sum is a sum of givens, every sum is constructed from a unique plurality of givens. So we have an extensional criterion of identity for sums: two sums are identical if and only if they are constructed from the same givens. In other words, the sum constructor is injective on pluralities of givens.

In Appendix E, we provide a proper formulation of the extremal clause as an induction principle. We also prove that (A30) is equivalent to this induction principle in the special case of mereological sums. The upshot is that, for present purposes, (A30) can be used as a formulation of the desired extremal clause.

We adopt the following definition of a predicate '$\leq$', where the intended meaning of '$x \leq y$' is that x is a (mereological) part of y:

(D28) $x \leq y \ \leftrightarrow \ \exists yy(\textsc{Sum}(y : yy) \wedge x \prec yy)$

So to be a part of an object is to a member of a plurality from which the object can be built using the sum constructor. This definition is analogous to the definition of set-theoretic membership (Section 9.9). The adequacy of both definitions will be vindicated by the ability to recover in the CCT key axioms of set theory and mereology (Section 10).

Remark. It follows from $x \leq y$ that both x and y are sums. This can be seen as follows. If $x \leq y$, by definition there are yy such that $\textsc{Sum}(y : yy)$ and $x \prec yy$. Since $\textsc{Sum}(y : yy)$, y is a sum. Since $x \prec yy$ and yy is an input to the sum constructor, x is summable. Only individuals are summable, so x is an individual, i.e. a given or a sum. This implies that x is a sum, since it follows from Collapse that givens are sums.

Note that (A30) is a natural generalization of, and hence stronger than, the claim that every given is a mereological atom (relative to the above definition of parthood). Define an atom to be an individual whose only part is itself:

(D29) $\textsc{Atom}(x) \leftrightarrow \textsc{IsSum}(x) \wedge \forall y(y \leq x \rightarrow y = x)$

As we explain and prove in Appendix E, the claim that every given z is an atom can be formalized as:

$$\textsc{Given}(z) \wedge \forall x(x \prec xx \rightarrow \textsc{Given}(x)) \wedge \textsc{Sum}(z : xx) \rightarrow z \prec xx \wedge \forall x(x \prec xx \rightarrow x = z)$$

That is, if a given is the sum of some givens, then "it is those givens", i.e. it is the only member of those givens. Thus, where this principle says that every *given* is a unique sum of givens, namely the sum of itself, (A30) says that every mereological object, *whether a given or not*, is a unique sum of givens.

Note. Various generalizations will eventually be needed. First, we will want to allow the givens to come equipped with a mereological structure. This will involve exchanging (A30) for some other axiom(s). It will also involve changing the definition of mereological sum. Second, we will want to add "decomposers", which construct new parts of already existing objects. The result is a broader conception of what can exists at successor stages.

An important feature of the CCT is that it enables us to derive standard mereological axioms. More precisely, we can prove that the sums, ordered by $\leq$, form an atomistic General Extensional Mereology (AGEM) whose atoms are precisely the givens. Details can be found in later sections (10.3 and 10.4).

9.11 Left and right constructors

The left and right constructors, in combination with the pair constructors, build pairs in a symmetrical way to other constructors. They do so without the need for order. At present, pairs are built in two stages using these types of constructor. We are currently working on replacing this approach with a simpler, direct construction process that treats pairs as derived objects.

The left and right constructors prepare objects to become inputs for the pair constructor. The general picture is the following. A pair has a left coordinate and a right coordinate. In the CCT, inputs to constructors are pluralities, which have unordered members. So the pair constructor needs some information to select the member of the input plurality that will become the left coordinate and

the member that will become the right coordinate. We supply this information by constructing two kind of *sui generis* position objects: left objects and right objects.

The construction of left and right object is is constrained in two ways. The first is the size of the input plurality:

(A31)

$$\forall xx \forall x((\textsc{Left}(x : xx) \ \vee \ \textsc{Right}(x : xx)) \ \rightarrow \\ \forall y \forall z(y \prec xx \ \wedge z \prec xx \ \rightarrow y = z))$$

(Any input plurality to the left or right constructor has no more than one member.)

Since pluralities are not empty, this axiom ensures that the left and right constructors apply only to singleton pluralities. As a result, the single member of the singleton plurality used as input becomes a left or right object.

The second constraint concerns the sole member of the input plurality: it must be "positionable". We define this property as follows:

(D30)

$$\textsc{Positionable}(x) \ \leftrightarrow \\ (\textsc{IsSet}(x) \ \vee \ \textsc{Individual}(x) \ \vee \ \textsc{IsPair}(x) \ \vee \ \textsc{Trace}(x))$$

Note. It follows that position objects are not themselves positionable. This is a manifestation of their supporting role: position objects mark prior objects of interest as left or right so that they can be used, so marked, in the construction of a pair. The position object themselves are not objects of interest for this construction. As a result, the left and right constructors cannot be iterated. For example, there is no "left left object".

Once we have defined what counts as "positionable", we sanction the second requirement by means of an axiom.

(A32)

$$\forall xx \forall x((\textsc{Left}(x : xx) \ \vee \ \textsc{Right}(x : xx)) \ \rightarrow \\ \forall z(z \prec xx \ \rightarrow \textsc{Positionable}(z)))$$

(If some things construct a left or right object, then each of them is positionable.)

Every appropriate singleton plurality is eventually used to construct a left object and a right object:

(A33)

$$\forall xx(\forall y \forall z(y \prec xx \ \wedge \ z \prec xx \ \rightarrow \ y = z) \wedge \\ \forall z(z \prec xx \rightarrow \ \textsc{Positionable}(z)) \rightarrow \\ \exists x_1 \textsc{Left}(x_1 : xx) \ \wedge \ \exists x_2 \textsc{Right}(x_2 : xx))$$

(If xx is singleton plurality whose member is positionable, then a left object and a right object with input xx exist.)

Remark. As remarked in Section 9.9, one can reason from the axioms of our theory to the conclusion that the elements of a set exist at an earlier stage than the set itself. An analogous reasoning yields that the sole member of the plurality from which a position object is constructed exists at an earlier stage than the position object itself. Formally, we have:

$$\forall x(x@t \wedge (\textsc{Left}(x : xx) \ \vee \ \textsc{Right}(x : xx)) \rightarrow \exists s(s \lhd t \wedge xx@@s))$$

Like sets, position objects strictly depend on their input pluralities. So the left and right constructors are injective:

(A34) $\textsc{Injective}(c_{\text{left}}) \ \wedge \ \textsc{Injective}(c_{\text{right}})$

(Position types are injective. That is, a position object, left or right, is constructed from at most one plurality.)

In combination with the functionality of the generic constructor, this axiom entails that position objects are extensional: two position objects of the same type are identical if and only if they are constructed from the same plurality.

9.12 Pair constructor

Our current approach to pairs avoids an explicit order, relying on left object and right objects to mark positions in a pair. We recover the paired objects from the inputs that generated the position objects used to construct the pair. For example, suppose we want to pair a and b in this order, choosing left and right to represent the first and second coordinate respectively. Then we use the singleton plurality of a to construct a left object a', and we use the singleton plurality of b to construct a right object b'. Finally, we build a pair from the plurality of a' and b'. We recover the first coordinate by considering the sole member of the plurality that constructed the left object a'. This yields a, as desired. We recover the second coordinate similarly.

This approach requires that the input plurality to the pair constructor have exactly two members, a left object and a right object. We formalise the requirement using the following axiom:

(A35)

$$\begin{aligned}&\forall xx\forall x(\textsc{Pair}(x : xx) \ \rightarrow \\ &\quad \exists y\exists z(y \prec xx \ \wedge \ \textsc{IsLeft}(y) \ \wedge z \prec xx \ \wedge \ \textsc{IsRight}(z) \wedge \\ &\quad \forall w(w \prec xx \ \rightarrow \ (w = y \ \vee \ y = z))))\end{aligned}$$

(Any input plurality to the pair constructor has exactly two members: a left object and a right object.)

Thus, "pairable" objects are left and right objects. As noted, however, not every plurality of pairable objects can be an input to the pair constructor. Pairable object objects must combine in the particular way specified by the above axiom: exactly one left object and one right object.

There are additional axioms. Every appropriate plurality is eventually used to construct a pair:

(A36)

$$\begin{aligned}\textsc{IsLeft}(x) \wedge \textsc{IsRight}(y) \rightarrow \\ \exists zz(\forall z(z \prec zz \leftrightarrow (z = x \vee z = y)) \wedge \\ \exists z\textsc{Pair}(z : zz))\end{aligned}$$

(For every left object x and right object y, a pair with input x and y exists.)

Remark. In analogy with the case of sets and position objects, we can prove from the axioms of the CCT that the objects used to build a pair exist at earlier stages than the pair itself. The claim is formalised as follows:

$$\textsc{Pair}(z : xx) \wedge z@t \rightarrow \forall x(x \prec xx \ \rightarrow \ \exists s(s \lhd t \ \wedge \ x@s))$$

Moreover, like sets and position objects, pairs strictly depend on their input pluralities. So their constructor is injective:

(A37) $\textsc{Injective}(c_{\text{pair}})$

(The type pair is injective. That is, a pair is constructed from at most one plurality.)

As before, the functionality of the generic constructor and the injectivity of the specific constructor entail extensionality. Here we have that two pairs are identical if and only if they are constructed from the same plurality.

Note. In general, the present approach to pairs requires two stages to build a pair. We start with some objects a and b. At a subsequent stage we turn them into position objects. It follow from (A34) that these position objects are distinct if a and b are distinct. The position objects are then used as inputs to the pair constructor at a later stage. There is some analogy with the construction of pairs as set-theoretical objects according to Kuratowski's definition. We start with a and b. We then build $\{a\}$ and $\{a, b\}$. Finally, we build the desired set $\{\{a\}, \{a, b\}\}$, which stands for the pair $\langle a, b\rangle$.

Note. How can our approach be extended from pairs to tuples of arbitrary length? A natural option is to add a new constructor for each coordinate of the desired tuples. This would allow us to construct position objects corresponding to the relevant coordinates. In our setting, the effect of new constructors can be obtained by specialising the generic constructor through parameters. For these parameters, one could use, for example, set-theoretic renderings of the natural numbers associated with the target coordinates. So the number one

(say $\{a\}$ for some conventionally chosen object a) would be used as the type of constructor for the first coordinate of a tuple, the number two ($\{\{a\}\}$) would be used as the type of constructor for the second coordinate, and so on. Once the appropriate position objects have been built, we obtain a tuple by applying the tuple constructor to such objects. This idea can be extended to infinite sequences simply by using a larger stock of parameters (e.g. a representation of the ordinals).

An alternative option is to develop a general theory of sequences, perhaps based on two constructors: one constructor for turning an object into the corresponding sequence of length one and another for concatenating two sequences. We leave this for future work.

9.13 Union constructor

Among the constructors countenanced by the CCT, the union constructor has a special status. It is the only *derived* constructor, in the following sense. The objects constructed by the union constructor are sets, which can also be constructed using the set constructor. By contrast, not all sets can be generated by means of the union constructor alone. For examples, singleton sets of givens cannot be so constructed. Singleton sets are effectively "givens" for the union constructor. The upshot is that the union constructor could be dispensed with. However, pragmatic concerns as well as faithfulness to the target TLOs recommend that the union constructor be included alongside the other constructors.

A set z is the union of the sets xx if and only if every member of xx is a set and something is an element of z just in case it is an element of some set in xx. So the union of the sets xx is the set constructed from the plurality of elements of members of xx. We capture the relation between xx and their members by means of a definition:

(D31)

$$\begin{aligned}&\textsc{Unionise}(xx, yy) \leftrightarrow \\ &\qquad \forall y(y \prec yy \rightarrow \textsc{IsSet}(y)) \wedge \\ &\qquad \forall x(x \prec xx \leftrightarrow \exists y(y \prec yy \wedge \exists zz(x \prec zz \wedge \textsc{Set}(y : zz))))\end{aligned}$$

That is, $\textsc{Unionise}(xx, yy)$ holds if and only if yy are some sets whose elements, combined, are all and only xx.

This relation between pluralities is used in the following axiom, which describes the link between the union constructor and the set constructor:

(A38) $\textsc{Union}(z : xx) \leftrightarrow \exists yy\ (\textsc{Set}(z : yy) \wedge \textsc{Unionise}(yy, xx))$

(z is the union of xx if and only if z is the set of some yy that "unionise" xx.)

A consequence of the axioms is that if $\textsc{Union}(z : xx)$, then each member of xx is a set. So "unionable" objects are sets.

Remark. Since unions are sets, their identity conditions are inherited from those of sets and need not be specified separately. However, it may be useful to describe an appropriate identity criterion that could be adopted in contexts where the union constructor is not treated as derived:

$$\textsc{Union}(z_1 : xx_1) \wedge \textsc{Union}(z_2 : xx_2) \rightarrow$$
$$(\exists yy(\textsc{Unionise}(yy, xx_1) \wedge \textsc{Unionise}(yy, xx_2)) \leftrightarrow z_1 = z_2)$$

(The unions constructed from xx_1 and xx_2 are identical if and only if the same things unionise both xx_1 and xx_2.)

Remark. We want to think of a union as dependent on the sets from which it is constructed. Nevertheless, we do not add an axiom to that effect. Once we identify unions with sets, a union exists at a stage if and only if the set to which it is identical exists at that stage. This means that a union can exist before some sets that could construct it exist. For example, if we have two givens a and b, we could form the set $\{a, b\}$ before forming the singletons $\{a\}$ and $\{b\}$. So the union of these singletons can exist before the singletons do.

Given our setup, however, we can let traces express the idea that a union is generated after the sets that construct it. Consider the union z constructed from the sets xx. We introduce traces of the construction of z from xx at the first stage t such that z exists at t and xx exist at a stage prior to t. This simulates that z has been constructed from xx at t. Returning to our example, we record the construction of $\{a, b\}$ as the union of $\{a\}$ and $\{b\}$ at the first stage where $\{a, b\}$ exists after $\{a\}$ and $\{b\}$ have been constructed.

9.14 Traces

The next set of axioms concerns traces. These axioms ensure that whenever certain constructions are effected, there are objects that encode relevant information about the construction process. The name 'traces' emphasizes that the role of these objects is to log information about constructions. Let us consider an example. Suppose the set z is constructed from xx, i.e. $\textsc{Set}(z : xx)$. Then, for *each member* x of the input plurality xx, there is a trace w that stores the following information about the underlying construction process:

(i) the output (the object constructed);

(ii) the fact that x is one of the inputs;

(iii) the type (set);

Using our primitive predicates, the information about the example would be formalised as follows:

(i) $\textsc{HasOutput}(w, z)$

(ii) $\textsc{HasInput}(w, x)$

(iii) $\textsc{HasType}(w, c_{\text{set}})$

Thus a trace can be seen as a reification of the constructional relation between composite and component, i.e. between the output and each member of the input to the construction. Suppose that we use the plurality of a and b, two distinct objects, to build the set z. Then two traces w_1 and w_2 emerge from this construction. Both traces have output z and type c_{set}. However, while w_1 has input a, w_2 has input b. In general, the information that z is a set constructed from some objects xx can be obtained by consulting all the traces that have the set z as an output.

We characterise the traces by specifying the behaviour of the primitive predicates associated with them. First, we require that an object always carries the threefold information just described or does not carry any information of this kind:

(A39)
$$\forall w((\exists z \textsc{HasOutput}(w, z) \leftrightarrow \exists x \textsc{HasInput}(w, x)) \wedge (\exists z \textsc{HasOutput}(w, z) \leftrightarrow \exists y \textsc{HasType}(w, y)))$$

(Any object has an output if and only if it has an input if and only if it has a type.)

Thus we can define a trace simply as an object that carries information about an output:

(D32) $\textsc{Trace}(w) \leftrightarrow \exists z \textsc{HasOutput}(w, z)$

We now lay out the identity conditions for traces:

(A40)
$$\begin{aligned}&\textsc{Trace}(w_1) \wedge \textsc{Trace}(w_2) \rightarrow \\ &\forall x_1 \forall x_2 \forall y_1 \forall y_2 \forall z_1 \forall z_2((\textsc{HasOutput}(w_1, z_1) \wedge \textsc{HasOutput}(w_2, z_2) \wedge \\ &\quad \textsc{HasInput}(w_1, x_1) \wedge \textsc{HasInput}(w_2, x_2) \wedge \\ &\quad \textsc{HasType}(w_1, y_1) \wedge \textsc{HasType}(w_2, y_2)) \rightarrow \\ &\qquad (z_1 = z_2 \wedge x_1 = x_2 \wedge y_1 = y_2 \leftrightarrow w_1 = w_2))\end{aligned}$$

(Two traces are identical if and only if they have the same output, input, and type.)

Remark. Together with (A39), these identity conditions entail that any trace carries information about exactly one output, exactly one input, and exactly one type.

The next step is to ensure that traces are introduced at the correct stages. That is, when an object is constructed at a stage—or, as in the case of union, it is *treated* as constructed at a stage—the appropriate traces are also introduced at that stage.

(A41)

$$(\textsc{Construct}(z : xx, y) \ \wedge \ z@t \ \wedge \ \exists s(s \lhd t \ \wedge \ xx@@s)) \rightarrow$$
$$\forall x(x \prec xx \ \rightarrow \ \exists w\big(w@t \wedge$$
$$\textsc{HasOutput}(w, z) \ \wedge \ \textsc{HasInput}(w, x) \ \wedge \ \textsc{HasType}(w, y)))$$

(Suppose z is constructed from xx with type y, z exists at t, and xx exist at some stage before t. Then we also require at t the existence of appropriate traces. That is, for every member x of xx, there is at t a trace recording that the construction process has output z, input x, and type y.)

Remark. The condition

$$\exists s(s \lhd t \ \wedge \ xx@@s)$$

in the antecedent of (A41) is redundant for sets, position objects, and pairs. This is because, as remarked above, constructions of these types require the members of the input plurality to exist at an earlier stage than the constructed object. In the case of sums or unions, however, the condition is not redundant.

Some examples might help illustrate the effect of this condition. Let us use '+' to indicate the result of summing some objects. And let us indicate the plurality of $x_1, ..., x_n$ as $[x_1, ..., x_n]$. So, for example, $\textsc{Sum}(a+b : [a, b])$. Suppose that the initial stage, stage 0, comprises three givens a, b, and c. Suppose further that at stage 1 we have the following individuals: a, b, c, $a + b$, $a + c$, $b + c$, and $a + b + c$. Consider the construction $\textsc{Sum}(a + b + c : [a + b, c])$. While $a + b$ and c exist at stage 1, no traces of this construction is introduced at stage 1. This is because $a + b$ does not exist at the previous stage. The traces of $\textsc{Sum}(a+b+c : [a+b, c])$ are introduced at stage 2, when the conditions laid out in the antecedent of (A41) are met. By contrast, the traces of $\textsc{Sum}(a + b + c : [a, b, c])$ exist at stage 1, since $a + b + c$ exists at that stage and each member of $[a, b, c]$ exists at the previous stage. This distribution of traces expresses that the construction $\textsc{Sum}(a + b + c : [a + b, c])$ takes place at stage 2, while the construction $\textsc{Sum}(a + b + c : [a, b, c])$ takes place at stage 1.

Something similar applies in the case of unions. Suppose the only set constructed at stage 1 is $\{a, b, c\}$, whereas $\{a, b\}$ and $\{c\}$ are constructed at stage 2. While $\{a, b, c\}$ is the union of $\{a, b\}$ and $\{c\}$, it exists prior to them. In this scenario, the traces of $\textsc{Union}(\{a, b, c\} : [\{a, b\}, \{c\}])$ are introduced at stage 3, namely, the first stage such that $\{a, b, c\}$ exists at it and both $\{a, b\}$ and $\{c\}$ exist prior to it. This expresses that the construction $\textsc{Union}(\{a, b, c\} : [\{a, b\}, \{c\}])$ takes place at stage 3.

We have given sufficient conditions for the existence of traces, ensuring that construction processes are accompanied by the appropriate traces. We now want to give necessary conditions, ensuring that there are no more traces than required. This is done by an axiom stating that the only traces existing at a stage are those recording constructions (basic or derived) using objects from previous stages:

(A42)

$$\textsc{Trace}(w) \wedge w@t \rightarrow$$
$$\exists z \exists xx \exists x \exists y \exists s(\textsc{Construct}(z : xx, y) \wedge z@t \wedge xx@@s \wedge s \lhd t \wedge$$
$$x \prec xx \wedge \textsc{HasOutput}(w, z) \wedge \textsc{HasInput}(w, x) \ \wedge \ \textsc{HasType}(w, y))$$

(If a trace w exists at a stage t, then it must appropriately record some type of construction of an object existing at t from an input existing at some stage before t.)

Remark. Axiom (42) implies that there are six kinds of traces, corresponding to the five basic constructors (set, sum, left, right, and pair) and to the derived constructor of union.

9.15 Maximal extension of a stage

A stage t is a maximal extension of a stage s if and only if t contains every object that is constructed from objects that exist at s. This is formalised by the next definition:

(D33) $\textsc{Max}(s, t) \ \leftrightarrow \ \forall x(\textsc{ConstrFrom}(x, s) \rightarrow x@t)$

Remark. The axioms for traces ensure that the appropriate traces exist at maximal stages.

We assume maximality in the following sense:

(A43) $\forall s \exists t \, \textsc{Max}(s, t)$

(Every stage has a maximal extension.)

Given any stage, the axioms for the specific constructors ensure that the objects existing at that stage are used for every applicable construction. For example, axiom (A23) states that for every plurality xx of settable objects existing at a stage, the set of xx is constructed. Maximality requires the objects constructed from a given stage to exist together at some stage.

Remark. A maximal extension strictly succeeds the stage whose maximal extension it is: $\textsc{Max}(s, t) \rightarrow s \lhd t$.

Note. The theory implies that no new sums are constructed after the first maximal extension of the initial stage. This is because, as observed in Section 9.10, two sums are identical if and only if they are sums of the very same pluralities of givens. So once every plurality of givens has been summed, every sum that is constructed must be identical to one that has already been constructed.

Remark. Consider the statement that every successor stage is also a maximal extension:

$$\textsc{Succ}(s, t) \rightarrow \textsc{Max}(s, t) \qquad (*)$$

This statement requires, in effect, that all constructional possibilities admitted by the theory be realised as soon as possible. Without the statement, the construction process may be slowed down: some constructional possibilities available at a stage maybe be realised before other constructional possibilities also available at that stage. For example, one may start by constructing mereological sums of givens before constructing sets of givens. Thus (*) provides a switch that makes the construction process as fast as possible.

9.16 Induction on the construction of objects

The possibility of induction on the construction of objects is an appealing feature of the constructional approach, a feature that might non be available in a non-constructional setting. In Section 9.10, we used an induction on the construction of sums to show that every sum is a sum of givens. As noted, this specific form of induction relies essentially on the fact that givens and sums are individuals and that only individuals are summable. So it has no obvious analogue in cases of constructors that permit as inputs also objects built from other constructors.

However, we have the resources to prove from our axioms a general form of induction on the construction of objects. Assume that

(i) every given is φ, and

(ii) if every input to any constructor is φ, then the constructed object and its associated traces are φ.

Then everything that isn't a stage is φ.

We prove this as follows. Assume, for *reductio ad absurdum*, that the assumptions hold but there is a non-φ that isn't a stage. By the well-foundedness of the stages, there is a least stage, s, at which such an object x exists. By the cumulativity of the stages, s cannot be a limit stage but must be a successor. By (A18) and (A42), x must either be constructed from some objects yy available at the preceding stage or be a trace associated to a construction effected at s. If x is constructed from yy, then each of yy is φ, since s is the least stage at which a non-φ exists. Then it follows from assumption (ii) that x too is φ, contradicting our assumption that x is non-φ. If x is a trace, then it emerges at s from a construction whose inputs are available at the preceding stage. Then, again, it follows from assumption (ii) that x too is φ, contrary to our assumption. We conclude that everything that isn't a stage is φ. The proof appeals to stages and their well-foundedness, which highlights a benefit of the stage-theoretic framework.

10 Derivation of set theory and mereology

10.1 Axioms of set theory

This section presents key set-theoretic axioms that can be recovered within the CCT. Our exposition follows Florio and Linnebo 2021, Section 4.7. Standard

set theory, Zermelo-Fraenkel set theory with the Axiom of Choice (ZFC), is a theory of pure sets, formulated in the language of first-order logic containing only one non-logical predicate, '$\in$' for membership. (All other set-theoretic notions are defined in terms of this single predicate.) The CCT describes, additionally, entities that are not sets. So the appropriate set-theoretic axioms in our context make room for such entities. This can be done by suitable restrictions of the quantifiers in the axioms of ZFC. The result is similar to what is often known as ZFCU, namely, ZFC with *urelements*. The traditional role of urelements is played here by the givens alone, not by all entities that aren't sets. We recognise various non-sets in addition to the givens. Indeed, some of these non-sets arise arbitrarily high in the hierarchy of stages, which implies that they do not form a set. By contrast, there is a set of all gives. This corresponds to the traditional axiom that the urelements form a set.

Another important difference between our target set-theoretic axioms and the axioms of ZFC concerns the empty set. ZFC requires the existence of an empty set, which is unavailable in the CCT. So we must also modify the axioms of ZFC to avoid any commitment to the empty set.

A third difference arises from the fact that not all objects in the CCT are settable. For example, position objects cannot be used to form sets. So, in formulating the target set-theoretic axioms, we must take care to respect this fact.

Below we list the target axioms. We show in the next section how to derive them from the axioms of the CCT. Recall that the we have provided a constructional definition of $\in$ in Section 9.9: $x \in y$ if and only if y is a set constructed from some plurality that has x as a member. This definition is operative in the following axioms.

(Extensionality) $\textsc{IsSet}(x) \wedge \textsc{IsSet}(y) \rightarrow (\forall u(u \in x \leftrightarrow u \in y) \rightarrow x = y)$

(Coextensive sets are identical.)

(Pairing)

$$\textsc{Settable}(x) \wedge \textsc{Settable}(y) \rightarrow \exists z(\textsc{IsSet}(z) \wedge \forall u(u \in z \leftrightarrow u = x \vee u = y))$$

(Every two settable objects have a pair set.)

(Union)

$$\textsc{IsSet}(x) \wedge \forall y(y \in x \rightarrow \textsc{IsSet}(y)) \rightarrow \exists z(\textsc{IsSet}(z) \wedge \forall u(u \in z \leftrightarrow \exists w(u \in w \wedge w \in x))$$

(For every set x whose elements are sets, there is a set y whose elements are precisely those objects that are elements of some element of x.)

(Powerset)

$$\textsc{IsSet}(x) \rightarrow \exists y(\textsc{IsSet}(y) \wedge \forall u(u \in y \leftrightarrow \textsc{IsSet}(u) \wedge \forall w(w \in u \rightarrow w \in x))$$

(Every set has a powerset.)

(Infinity)

$$\exists x \exists y(\textsc{Given}(y) \ \wedge \ y \in x \ \wedge \ \forall z(z \in x \ \rightarrow \ \exists u(\forall w(w \in u \ \leftrightarrow \ w = z) \ \wedge \ u \in x)))$$

(There is an infinite set. More specifically, there is a set x with a given as an element and such that, whenever z is an element of x, so too is its singleton $\{z\}$.)

(Separation)

$$\exists y(y \in x \ \wedge \ \varphi(y)) \rightarrow \exists z \forall u(u \in z \leftrightarrow u \in x \ \wedge \ \varphi(u))$$

(For any set x and any condition φ satisfied by at least one element of x, there is a set of precisely those elements of x that satisfy φ.)

(Foundation)

$$\textsc{IsSet}(x) \ \rightarrow \ \exists y(y \in x \wedge \neg \exists z(z \in y \wedge \ z \in x))$$

(Every set x has an element that is disjoint from x.)

(Replacement)

$$\textsc{IsSet}(x) \ \wedge \ \forall y \forall z(\psi(y,z) \ \rightarrow \ \textsc{IsSettable}(z) \wedge \forall u(\psi(y,u) \ \rightarrow \ u = z)) \wedge \exists y \exists z(y \in x \ \wedge \psi(y,z)) \ \rightarrow \ \exists w \forall u(u \in w \ \leftrightarrow \ \exists y(y \in x \ \wedge \ \psi(y,w)))$$

(For every set x and functional condition ψ whose image includes only settable objects, if some element of x bears ψ to some object, then there is a set of precisely those objects that are borne ψ by some element of x.)

This axiom is based on a simple and intuitive idea. Consider any set. For each of its elements, choose either to keep this element or to replace it with some other settable object. Then the resulting collection is also a set.

(Choice)

$$\textsc{IsSet}(x) \ \wedge \ \forall y(y \in x \ \rightarrow \ \textsc{IsSet}(y)) \wedge \forall y \forall z(y \in x \ \wedge \ z \in x \ \wedge \ y \neq z \ \rightarrow \ \neg \exists u(u \in y \ \wedge \ u \in z)) \ \rightarrow \ \exists w(\textsc{IsSet}(w) \ \wedge \ \forall y(y \in x \ \rightarrow \ \exists u_1(u_1 \in y \ \wedge \ u_1 \in w \ \wedge \forall u_2(u_2 \in y \ \wedge \ u_2 \in w \ \rightarrow u_2 = u_1))))$$

(Every set x of non-empty disjoint sets has a choice set, that is, a set containing precisely one element of each element of x.)

An example due to Russell might be useful to understand the Axiom of Choice. Suppose you have infinitely many pairs of shoes. Then it is easy to define a set containing precisely one member of each pair, namely, the set of left shoes. What if you have infinitely many pairs of socks where the two members of each pair are indistinguishable? Then we are unable to define a set containing precisely one member of each pair. The Axiom of Choice tells us that such a set exists, irrespective of our ability to define it.

10.2 Derivation of the axioms of set theory

In this section, we show how the axioms of set theory presented in the previous section can be recovered from the CCT. We proceed in the same order.

Extensionality. We want to show that coextensive sets are identical. Consider two coextensive sets x and y. Then x and y are constructed from the same plurality of objects. Thus, by the functionality of the set constructor, we have $x = y$, as desired.

Pairing. We want to show that every two settable objects have a pair set. Consider two settable objects x and y. Since stages are cumulative, there is a stage at which each of the two objects exists. By plural comprehension, we can form the plurality of x and y, which exists at that stage. Thus, by (A23), there is the set of x and y, which is our desired pair set.

Union. We want to show that for every set x whose elements are sets, there is a set y whose elements are precisely those objects that are an element of some element of x. Given a set x existing at t, we can find yy at some earlier stage s from which x is constructed. For any set among yy, there are some objects from which this set is constructed. These objects exist prior to s. Thus every element of some member of yy also exists at s. By plural comprehension, the plurality of such elements exists at s. Thus, by (A23), this plurality forms a set. This is the desired union set.

Powerset. We want to show that every set has a powerset. Let x be a set whose elements are yy. If x exists at stage, yy also exist at that stage. It follows from (A23) that for every subplurality zz of yy, the set of zz exists. The maximality axiom (A43) yields a stage t at which every set constructed from a subplurality of yy exists. Consider the plurality ww of such sets. Since this plurality exists at t, we can use (A23) to conclude that there is the set of ww, which is the desired powerset.

Infinity. Recall that we adopted (A25), a set-theoretic version of the principle that there is an infinite stage. The axiom states that there is a plurality existing at a stage such that it has a given as a member and, whenever x is a member of it, so too is the singleton of x. The set of this plurality exists by (A23), and it is infinite, i.e. it satisfies the closure conditions just mentioned.

Separation. We want to show that for any set x and any condition φ satisfied by at least one element of x, there is a set of precisely those elements of x that satisfy φ. Given a set x existing at t, there are yy at an earlier stage s from which x is constructed. By plural comprehension, there is a subplurality zz of those members of yy satisfying φ. By applying the set constructor to zz, we obtain the set of precisely those elements of x that satisfy φ.

Foundation. We want to show that every set x has an element that is disjoint from x. Given a set x, consider its elements yy. By the well-foundedness of the stages, there is a least stage s at which some element of x can be found. Let y be any such element. If y is not a set, then y has no member and is therefore disjoint from x. If y is a set, then every element of y exists prior to s and hence prior to all elements of x, since these do not exist prior to s. It follows that no element of y is an element of x. Thus y and x are disjoint, as desired.

Replacement. We want to show that for every set x and functional condition ψ whose image includes only settable objects, if some element of x bears ψ to some object, then there is a set of precisely those objects that are borne ψ by some element of x. Let x be a set existing at s, and let ψ be a functional condition of the kind just described. The elements of x also exist at s. Suppose at least one of them bears ψ to some object. Since ψ-images are settable objects and hence not stages, we can apply axiom (A11), a stage-theoretic version of Replacement, to find a stage t at which every ψ-image of some element of x exists. By plural comprehension, the plurality of such images exists. Since every member of this plurality is settable and exists at t, it follows from (A23) that a corresponding set exists. This is the set of precisely those objects that are borne ψ by some element of x, as desired.

Choice. This axiom can be proved if we assume a corresponding choice principle for pluralities. Some natural formulations of the principle are schematic. In this context, we prefer the following formulation, which is expressed as a single statement. Let pp be a plurality of pairs (say, ordinary Kuratowski pairs). Each first coordinate x occurring in one of these pairs is associated with a unique plurality of objects, namely, all the objects yy such that the pair $\langle x, y\rangle$ is one of pp and its second coordinate y is one of yy. Thus, pp can be seen as representing a family of pluralities of objects, all the pluralities yy such that yy is associated in the way just described with some x occurring as a first coordinate in pp. We can now state the plural choice principle. Suppose pp represents a family of pairwise disjoint pluralities of objects. Then there is a plurality cc such that, for every x that occurs as a first coordinate in pp, there is a unique $y \prec cc$ such that $\langle x, y\rangle \prec pp$. (Then cc is known as a *choice plurality* for pp.) Now, this plural choice principle obviously implies the set-theoretic Axiom of Choice. To see this, let x be a set of non-empty disjoint sets. Suppose x exists at a stage s. Let pp be the plurality of all and only pairs $\langle u, v\rangle$ such that $u \in x$ and $v \in u$. The choice principle for pluralities ensures that there is a choice plurality cc for pp. Since every member of cc is an element of an element of x, cc must all exist

at s, if not before. Thus, (A23) yields a set c constructed from cc, which is easily seen to be a choice set for x.

10.3 Axioms of mereology

This section provides an exposition of the axioms of atomistic general extensional mereology (AGEM). In the next section, we show how to derive these axioms from those of the CCT.

Several equivalent axiomatisations of GEM can be found in the literature. The axioms adopted here are based on those proposed in Cotnoir and Varzi 2019 (see also Florio and Linnebo 2021, pp. 96–103). We could simply import these axioms and ensure they have the appropriate scope by restricting all their quantifiers to sums. As remarked in Section 9.10, it follows from $x \leq y$ that both x and y are sums. This enables us to simplify some of the axioms by dropping redundant quantificational restrictions.

First, the sums are a partial order with respect to $\leq$:

(Reflexivity) $\textsc{IsSum}(x) \to x \leq x$

($\leq$, restricted to sums, is reflexive.)

(Antisymmetry) $x \leq y \ \wedge \ x \leq y \ \to x = y$

($\leq$, restricted to sums, is antisymmetric.)

(Transitivity) $x \leq y \ \wedge \ y \leq z \ \to x \leq z$

($\leq$, restricted to sums, is transitive.)

Next there is an axiom scheme of unrestricted fusion. For every formula φ where x but neither y nor z occur free, we have:

(Unrestricted Fusion)

$$\forall x(\varphi(x) \to \textsc{IsSum}(x)) \ \wedge \ \exists x\varphi(x) \to \exists y(\textsc{IsSum}(y) \wedge \forall z(y \leq z \leftrightarrow \forall x(\varphi(x) \to x \leq z)))$$

(Suppose that every φ is a sum and there is at least one φ. Then there is a sum y such that for every z, $y \leq z$ if and only if, for every x that is φ, $x \leq z$. This means that there is a least upper bound, with respect to $\leq$, of the φs.)

The last axiom of GEM is a principle of complementation (called 'Remainder' in Cotnoir and Varzi 2019):

(Complementation)

$$\textsc{IsSum}(x) \ \wedge \ \textsc{IsSum}(y) \ \wedge \ \neg x \leq y \ \to \exists z(\textsc{IsSum}(z) \wedge \forall w(w \leq z \leftrightarrow (w \leq x \wedge \neg\exists u(\textsc{IsSum}(u) \wedge u \leq w \wedge u \leq y))))$$

(For any two sums x and y, if x is not part of y, there is a sum z whose parts are all and only the parts of x that do not overlap y. Two sums overlap if a sum is part of both. The sum z can be regarded as the relative complement of y with respect to x.)

We obtain the target theory, AGEM, by adding an axiom of atomicity. Recall the definition of atom from Section 9.10:

(D29) $\textsc{Atom}(x) \leftrightarrow \textsc{IsSum}(x) \wedge \forall y(y \leq x \rightarrow y = x)$

Then the axiom of atomicity states:

(Atomicity) $\textsc{IsSum}(x) \rightarrow \exists y(\textsc{Atom}(y) \wedge y \leq x)$

(Every sum has an atomic sum as a part.)

This completes the presentation of the axioms of AGEM.

10.4 Derivation of the axioms of mereology

In this section, we show how to recover the mereological axioms presented in the previous section. The result is that, in the CCT, the sums ordered by $\leq$ satisfy the axioms of AGEM, where the atoms are precisely the givens. An important role in the derivations of the axioms is played by the principle that two sums are identical if and only if they are constructed from the same givens (Section 9.10). We also rely on the following, useful fact linking parthood with inclusion among pluralities of givens.

Fact. $x \leq y$ if and only if the givens whose sum is x are among the givens whose sum is y.

Proof. We start with the right-to-left direction of the biconditional. Assume that x and y are the sums of gg and hh, respectively, where $gg \preccurlyeq hh$. Then by Leveling, y is also the sum of the plurality zz obtained from hh by replacing gg with their sum. But the sum of gg is x. So x is one of some objects (i.e. zz) whose sum is y. This means that $x \leq y$. We now show the left-to-right direction. Assume that $x \leq y$, that is, x is one of some objects yy whose sum is y. Let g be one of the givens whose sum is x. Then, by Leveling, x is also the sum of x and g. Let zz be the plurality obtained from yy by replacing x with x and g. By Leveling again, the sum of zz is y. Since g is one zz, it follows that g is one of the givens whose sum is y. Since g was chosen arbitrarily from the givens whose sum is x, we conclude that the givens whose sum is x are among the givens whose sum is y.

We now proceed to derive the axioms of AGEM.

Reflexivity. Write x as a sum of givens gg. Clearly, $gg \preccurlyeq gg$, which by the Fact establishes that $x \leq x$.

Anti-symmetry. Assume that $x \leq y$ and $y \leq x$. We want to show that $x = y$. Suppose x and y are the sums of givens gg and hh, respectively. By the Fact, we have that $gg \preccurlyeq hh$ and $hh \preccurlyeq gg$, which entails that $gg \approx hh$. It follows from the functionality of the generic constructor that $x = y$.

Transitivity. Assume that $x \leq y$ and $y \leq z$. We want to show that $x \leq z$. Suppose x, y, and z are the sums of givens gg, hh, and ii, respectively. By the Fact, we have $gg \preccurlyeq hh$ and $hh \preccurlyeq ii$. So $gg \preccurlyeq ii$. This means that $x \leq z$.

Unrestricted Fusion. Assume that every φ is a sum and there is at least one φ. We want to how that there is a least upper bound, with respect to $\leq$, of the φs. Since there is some φ, plural comprehension yields the givens that are part of some φ. By (A27), their sum y exists. Clearly, y is an upper bound of the φs. To show that y is the *least* upper bound, suppose z is another upper bound. Since every x satisfying φ is part of z, we must also have, by the transitivity of $\leq$ (established above), that every given that is part of some φ is part of z. Thus, every given from which y is constructed is part of z and, by the Fact, also one of the givens from which z is constructed. It now follows, again from the Fact, that $y \leq z$ and thus also that y is the least upper bound, as desired.

Complementation. Consider two sums x and y such that $x \not\leq y$. We want to show that there is a relative complement of y with respect to x, namely, a sum whose parts are all and only the parts of x that do not overlap y. By the Fact, $x \not\leq y$ entails that some given from which x is constructed is not among the givens from which y is constructed. We can form the corresponding plurality of givens and, by (A27), construct the sum z of all and only the givens that are part of x but not of y. It is routine to show that z satisfies the criterion for being the relative complement of y with respect to x. To see this, consider any sum w that is part of x but does not overlap y. By the Fact, this is so if and only if w is a sum of givens drawn from among the givens whose sum is z. Again by the Fact, the preceding condition is satisfied if and only if $w \leq z$, which proves our claim.

Atomicity. As already observed in Section 9.10, every sum is a sum of givens, which are atomic. This establishes the claim.

11 Consistency of the Core Constructional Theory

A key technical component of the project is showing the consistency of the CCT, axiomatised in Section 9. As is often the case, this is done by constructing a set-theoretic model in which every axiom of the theory is true. The actual model construction and consistency proof can be found in Appendix F. Here we provide a brief overview.

To construct a set-theoretic model, we obviously need to rely on some set

theory in the metalanguage. Suppose this set theory is X. We will then have proved that "the CCT is consistent relative to X", that is, if our set theory X is consistent, then so is the CCT. This makes it important to let X be some familiar set theory of whose consistency set theorists are highly confident. One such theory is ZFC. Our options concerning the choice of the theory X are either ZFC itself or some relatively minor variants.

- We can use a Morse-Kelley class theory (MK), which adds to ZFC a single layer of classes on top of all of the sets.
- We can use ZFC + an extra axiom stating that there exists an "inaccessible cardinal". In the literature on strong axioms of infinity, this is considered a very modest extension of ZFC. (It is, however, stronger than MK, but has the advantage of talking only about sets, with no need to add a notion of class.)
- We can use ZFC as our metatheory and prove the consistency of a slight weakening of the CCT obtained by restricting the axiom of plural comprehension. Instead of stating that any condition $\varphi(x)$ that has instances can be used to define a plurality of all φs, we state that there is a plurality of all φs at stage s, for any given stage s.
- We can drop the axiom of Replacement from the CCT and use ZFC to prove the consistency of the resulting theory.

In short, the CCT can be proved to be consistent assuming some very modest extensions of standard set theory ZFC, and even the need to consider extensions can be avoided by weakening non-essential parts of the CCT. In Appendix F, we pursue the first option, namely we show that the CCT is consistent relative to MK.

12 Conclusion

This project set out to provide rigorously established foundations for the IMF's TLO by formalising the CCO. This constructional approach to ontology unifies the core components of the target 4-dimensionalist TLOs, leading to a single, common foundation for future development. Using this unified core to underpin the IMF's TLO will significantly simplify and strengthen it.

The approach is novel, involving subtle and complex technical issues. Hence, in our initial work, we only aimed to show the feasibility of the approach and provide a transitional solution as a good base for both current and future work.

The feasibility of a unified approach has been established by producing a formalisation of the CCO and by giving a mathematical proof of its consistency. The formal theory, the CCT, unifies the three key domains—sets, sums, and pairs—using a stage theory. In this theory, objects emerge through the application of different constructors to objects existing at the initial stage or constructed at prior stages. This unification and its associated simplification

are apparent in the common development of the key domains, the common basis for the identity criteria of the objects in them, and the uniformity through which we capture other important commonalities and differences. We also demonstrate the strength of the CCT by recovering the axioms of standard set theory and mereology.

Future work within the NDTp and planned academic research will develop further our constructional approach, resolving outstanding subtle and complex technical issues. For example, we will explore the introduction of deconstructors to work along constructors, thus enabling the possibility of mereological gunk. We may also explore alternative formalisations of the constructional approach, such as dynamic formalisations and formalisations that eliminate stages in favour of their associated pluralities. In addition, the next phase of the project will examine strategies for an automated proof of consistency, tackling the issues raised by the presence of schemes in the CCT.

A Notions

A.1 List of key types and identity criteria

The table below provides an informal summary of the seven broad types of objects in the CCT together with their identity criteria. There are six types of objects obtained from the application of constructors: sets, individuals, left objects, right objects, pairs, and unions. In addition, there are traces, which emerge from the constructional process. As is clear from the description in the table, all these types of objects satisfy extensional identity criteria, which directly correspond to the identity criteria in the target TLOs (see Section C).

type	construction	identity criteria
sets	all sets are constructed by the set constructor	two sets are identical iff they are constructed from the same plurality (equivalently, iff they have the same elements)
individuals	individuals are either constructed by the sum constructor or are givens	two individuals are identical iff they are constructed from the same givens (equivalently, iff they have the same parts)
left objects	all left objects are constructed by the left constructor	two left objects are identical iff they are constructed from the same plurality
right objects	all right objects are constructed by the right constructor	two right objects are identical iff they are constructed from the same plurality
pairs	all pairs are constructed by the pair constructor	two pairs are identical iff they are constructed from the same position objects (i.e. from the same plurality of left object and right object)
unions	all unions are constructed by the derivative union constructor	two unions are identical iff they are identical as sets (i.e. the sets to which they are equal are identical)
traces	all constructions have associated traces	two traces are identical iff they have the same type and record the same construction output and input

A.2 Constraints on inputs

Constructional processes are constrained in that not all kinds of pluralities can serve as inputs to constructions. Moreover, different constructors may place different restrictions on their inputs. The following table displays these constraints based on the broad types of members in the input plurality. For example, the set constructor admits as inputs only pluralities whose members are sets, individuals, pairs, unions, or traces. It is worth emphasising that there is no requirement that such pluralities be uniform: a plurality is admissible as long as each of its members is of an admissible type. So one can form the set $\{a, w\}$ where a is an individual and w is a trace.

type	settable	summable	positionable	pairable	unionable
sets	✓	✗	✓	✗	✓
individuals	✓	✓	✓	✗	✗
position objects	✗	✗	✗	✓	✗
pairs	✓	✗	✓	✗	✗
unions	✓	✗	✓	✗	✓
traces	✓	✗	✓	✗	✗

There are further constraints. Only singleton pluralities can be inputs to the construction of position objects. Moreover, the construction of a pair requires an input plurality containing a left object, a right object, and nothing else. As explained in Section 9.12, the paired objects are ultimately recovered from the inputs that generated the position objects used to construct the pair. So, while only position objects are pairable, the intended coordinates of pairs are, in effect, the positionable objects: sets, individuals, pairs, unions, and traces. Position objects function as mere intermediaries.

B Design choices

Given that we were working with a novel approach, we made it our first priority to show that a workable theory was possible. Hence, one could regard our first version of the CCT as a kind of minimum viable product (MVP). We accepted that, as a reasonable cost for early proof of concept, this would involve compromises leading to minor reductions in usability.

We needed to identify a safe approach, in particular with respect to the type of formalisation to use. The choice boiled down to stage theory or procedural postulationism (a theory outlined in Fine 2005). While the latter appears to be more attuned to constructionalism, it would require the development of significant logical machinery, which would have considerably delayed the production

of a first workable system. Accordingly, the safer option of stage theory, which has a better understood semantics, was chosen. If further work should prove it is feasible to develop a formal approach based upon procedural postulation, this alternative could be adopted later. Proposed improvements, including the choice of formalisation, are listed under future work in Appendix H.

As noted, our decision to develop an MVP led to minor reductions in usability. We now describe the most relevant, grouped by the associated kind of object. We plan to address these soon, implementing ideas we have been developing in the course of the project.

Let us call the informal approach of Partridge, Cesare, et al. 2017, Partridge, Mitchell, Loneragan, et al. 2019 and Partridge, Mitchell, Loneragan, et al. manuscript the *benchmark approach*. For sets, the formalisation required no compromise; in this respect, the CCT realises the benchmark approach. In other respects, there is some divergence, mostly as a result of choosing to develop an MVP.

The benchmark approach also adopts a mixture of composing and decomposing constructors. Specifically, it adopts a decomposer to construct individuals. The construction process starts with the pluriverse as a single given. All individuals are then obtained by applying a sum decomposer to the pluriverse. In contrast, the CCT uses a single form of constructor (composers) across the board, thus avoiding the complications of managing both composers and decomposers. This meant we had to adopt a different approach to individuals, one that relies on composers. We chose to start with mereological atoms as givens (informally, all the mereological atoms in the pluriverse) and to use a sum composer to construct the rest of the individuals. In effect, this amounted to swapping a sum decomposer for a sum composer. The resulting ontology is therefore as granular as the givens. It does not provide "access" to any parts that such givens might have or to whole-part relationships between the givens.

To build tuples, the benchmark approach uses a tuple constructor, assuming that this constructor can be applied to objects in a way that respects order and repetition. For this initial version of the CCT, we adopted a simpler approach that avoided the requirements for a formalisation of order and repetition. This is a natural and familiar approach to n-tuples: it starts with primitive pairs and opens up the possibility of coding n-tuples through nesting (see Remark in Section 9.12).

Four general objects are included in the benchmark approach and the target TLOs: set type, set-member type, super-sub-set type, and whole-part type. The decision has been made to incorporate them into the CCT as givens represented by special constants. As a result, these general objects can be inputs to constructors.

Given a constructional ontology, one might consider subontologies obtained from various possible selections of givens and constructors (Fine 1991). Investigating and comparing these subontologies might be useful. Partridge, Cesare, et al. (2017) have found that that it helps expose and explain the metaphysical structures of the candidate ontologies, and that it helps assess different architectural choices. The benchmark approach made use of "ontological sandboxes"

to allow the development and study of subontologies in isolation from the full constructional ontology. There was no requirement to replicate this feature in our MVP, hence no attempt was made to formalise it. However, the CCT has distinctive features whose explanatory benefits resemble those of ontological sandboxes. The CCT allows non-maximal stages, and it could be modified to allow multiple initial stages, each with its own givens. This allows for varieties of ontogeneses where constructors are exhausted in independent branches before merging—mimicking the behaviour of ontological sandboxes. For example, one could exhaust the sum constructor, constructing all individuals, before starting to apply the other constructors.

One question that emerges clearly in the formalisation is how far the construction process should go and, in particular, whether it should be extended into the transfinite. There is a requirement for the transfinite in the target TLOs, hence we have included it in the CCT.

C Supporting the IMF's selected TLOs

The CCO, and so the CCT, is intended to be the foundation of the IMF's TLO. This is a key part of the planned process of developing an FDM seed from a set of 4-dimensionalist top-level ontologies that best meet its technical requirements (West 2020). The table below shows how the CCO maps directly onto, and thereby supports, these selected TLOs.

CCT	BORO	IDEAS	ISO 15926	HQDM
"Object"	Objects	Thing	thing	thing
Individual	Elements	Individual	possible_individual	possible_individual
IsSet	Types	Type	class	class
IsPair	Tuples	tuple	relationship	relationship
Trace & c_{sum}	wholes-parts	wholePart	composition_of_individual	composition_of_individual
Trace & c_{set}	types-instances	typeInstance	classification	classification
Trace & c_{union}	super-sub-types	superSubType	specialization	specialization
"Position object"	tuple-places	tuplePlaces	end	end

Note that the CCO has a general Trace whose type can be recovered, whereas the other TLOs have individual types for each trace.

D CLAP background

Four key principles characterising identity conditions for constructed entities are singled out in Fine 2010, where they are expressed using constructors that take as inputs variable-length sequences of objects (including null sequences). Let Σ be a constructor of this kind. Then we have the following, independent principles.

(Collapse) $\Sigma(x) = x$

(The application of Σ to x is x.)

(Leveling) $\Sigma(..., \Sigma(x, y, ...), ..., \Sigma(u, v,), ...) = \Sigma(..., x, y, ..., u, v, ...)$

(Consider two sequences, one obtained from the other by replacing some subsequences with the results of applying Σ to those subsequences. Applying Σ to one sequence yields the same object as applying Σ to the other sequence.)

(Absorption) $\Sigma(..., x, x, ..., y, y, ...) = \Sigma(..., x, ..., y, ...)$

(Repetitions of an object in the input sequence of Σ are irrelevant to the result of the construction.)

(Permutation) $\Sigma(..., x, y, z, ...) = \Sigma(..., y, z, x, ...)$

(Changing the order of the objects in the input sequence of Σ is irrelevant to the result of the construction.)

Constructors and their outputs can be classified depending on which of these principles they satisfied. We call *CLAP profile* (from the initials of the principles' names) the particular combination of principles that characterise a given constructor and its outputs. Here are some important examples (Fine 2010):

- the CLAP profile of the set constructor is ~~C~~~~L~~AP, i.e. only Absorption and Permutation are satisfied;
- the CLAP profile of the sum constructor is CLAP, i.e. all four principles are satisfied;
- the CLAP profile of string concatenation is CL~~A~~~~P~~, i.e. only Collapse and Leveling are satisfied;
- the CLAP profile of the sequence constructor is ~~C~~~~L~~~~A~~~~P~~, i.e. none of the four principles is satisfied.

Another important example is the derived constructor of set-theoretic union:

- the CLAP profile of set-theoretic union is CLAP, i.e. all four principles are satisfied, as in the case of the sum constructor.

There are some differences between the framework of Fine's analysis and ours. Our approach relates to investigations of the target TLOs (Partridge, Cesare, et al. 2017, Partridge, Mitchell, Loneragan, et al. 2019, and Partridge, Mitchell, Loneragan, et al. manuscript) that sketched an informal foundation for the TLOs based on the following constructors and givens:

- constructors: set constructor, sum decomposer, sequence constructor, and the derived union constructor. (The sum decomposer is the inverse of the sum constructor; in other words, the direction of construction is reversed, which means that the decompose works from composites to components.)
- givens: a single object, the pluriverse (the fusion of all *individuals* in the sense of Section 9.5).

These investigations also identified a need for traces. In their setting, the following constraints emerged. The sum decomposer applies only to inputs that are individuals. Moreover, there are no "null applications" of constructors, thus no null set (that is, no empty set), null part, or null sequence—a constraint we imported. In this first formalisation, there were a number of hurdles to overcome. In the interest of getting a usable theory as quickly as possible, we made several design choices (see Appendix B) that resulted in a different set of constructor and givens:

- constructors: set constructor; sum constructor, replacing the use of the sum decomposer noted above; left constructor and right constructor; pair constructor, replacing the use of the sequence constructor noted above; set-theoretic union as a derived constructor.
- givens: six special objects used to indicate the types of construction.

We explain the motivations of these choices in Appendix B.

Further differences with respect to Fine's framework arise from our decision to adopt plural logic as the logical framework for our approach (see Appendix B). Our generic constructor takes a plurality as input, unlike the constructors involved in the four principles above, which take variable-length sequences of objects. So we need to interpret the four principles in our setting. We set out a way of doing this below. The discussion highlights some issues surrounding the formalisation of order and repetition, features to which pluralities are insensitive. We should also note that the use of pluralities as inputs naturally tracks the TLOs' requirement to exclude null applications of constructors.

Since we work with a generic constructor, we can interpret the four principles as properties of the parameter that specifies the type of construction. So we say that a type, such as set, does or does not satisfy Collapse, Leveling, Absorption, or Permutation. Let y be a type of construction. Then y satisfies the relevant principle if and only if it satisfies the corresponding condition below.

(Collapse) $\textsc{Construct}(z : xx, y) \wedge \forall x(x \prec xx \leftrightarrow x = u) \rightarrow z = u$

(If the generic constructor is applied using y to the singleton plurality of u, the outcome is u.)

(Leveling)

$$\begin{aligned}\textsc{Construct}(x_1 : xx_1, y) \ \wedge \ \textsc{Construct}(x_2 : xx_2, y) \ \wedge \\ \forall z_1(z_1 \prec xx_1 \ \rightarrow \ (z_1 \prec xx_2 \ \vee \\ \exists zz(zz \preccurlyeq xx_2 \ \wedge \ \textsc{Construct}(z_1 : zz, y)))) \ \wedge \\ \forall z_2(z_2 \prec xx_2 \ \rightarrow \ (z_2 \prec xx_1 \ \vee \\ \exists zz(zz \preccurlyeq xx_2 \ \wedge \ z_2 \prec zz \ \wedge \\ \exists z_3(z_3 \prec xx_1 \ \wedge \ \textsc{Construct}(z_3 : zz, y))))) \ \rightarrow \\ x_1 = x_2\end{aligned}$$

(Suppose x_1 is a constructed from xx_1 and x_2 is constructed from xx_2. If the following two conditions are satisfied, then x_1 is x_2:

1. every x_1 among xx_1 is either one of xx_2 or is constructed from some zz among xx_2;
2. every x_2 among xx_2 is either one of xx_1 or is one of some members of xx_2 that construct an object among xx_1.

In other words, two pluralities, one obtained from the other by replacing some subpluralities with the objects they construct, yield the same constructed object.)

(Absorption)

$$\begin{aligned}\textsc{Construct}(z_1 : xx_1, y) \ \wedge \ \textsc{Construct}(z_2 : xx_2, y) \ \wedge \ \exists u(u \prec xx_1 \ \wedge \\ \forall w(w \prec xx_2 \leftrightarrow w \prec xx_1 \vee w = u)) \ \rightarrow \\ z_1 = z_2\end{aligned}$$

(Repetitions of an object in the input plurality make no difference to outcome of the construction.)

(Permutation)

$$\begin{aligned}\textsc{Construct}(z_1 : xx_1, y) \ \wedge \ \textsc{Construct}(z_2 : xx_2, y) \ \wedge \ \exists uu \exists u \exists v(\\ \forall x(x \prec xx_1 \leftrightarrow z \prec uu \vee x = u \vee x = v) \ \wedge \\ \forall x(x \prec xx_2 \leftrightarrow x \prec uu \vee x = v \vee x = u)) \ \rightarrow \\ z_1 = z_2\end{aligned}$$

(Changing the order of the objects in the input plurality of makes no difference to the outcome of the construction.)

Remark. The principles of plural logic and the functionality of the generic constructor guarantee that these formulations of Absorption and Permutation holds for every type. So the CLAP profile of our constructors always includes A and P.

The formulations of Absorption and Permutation in Fine 2010 cover cases where two input sequences differ not by one but by multiple repetitions and changes in position. At least in the case of Absorption, one might try to strengthen the formulation of Absorption in our framework so that the presentations of the two plurality can differ by multiple (even infinitely many) repetitions:

$$\begin{aligned}\textsc{Construct}(z_1 : xx_1, y) \ \wedge \ \textsc{Construct}(z_2 : xx_2, y) \ \wedge \ \exists uu(uu \preccurlyeq xx_1 \wedge \\ \forall w(w \prec xx_2 \leftrightarrow w \prec xx_1 \vee w \prec uu)) \ \rightarrow \\ z_1 = z_2\end{aligned}$$

However, the resulting principle is still guaranteed by plural logic and the functionality of the generic constructor.

This is one of the areas where further research would be fruitful. In particular, it would be worthwhile to explore ways of adapting our framework, or using different frameworks, so as to make available constructors exhibiting the full range of CLAP profiles.

E Constructing via CLAP profiles

We would like an approach to constructed objects that is based entirely on their CLAP profile, as laid out in Fine 2010. In this appendix, we will largely follow the notation and terminology of this article rather than that of our exposition above. We make this choice for two reasons. First, Fine's notation affords greater economy and perspicuity for the purposes of our investigation in this appendix. Second, the modular nature of our approach ensures that this choice is benign. This technical appendix offers a mathematical investigation of the extremal clause in sense of Section 9.10. Its main aim is to justify our choice of (A30) as a simple and convenient formulation of the extremal clause for mereological sums.

We restrict our attention to cases where the principles of Absorption and Permutation are accepted. In all these cases, the argument of the constructor Σ can be taken to be just a plurality of objects. Which pluralities are eligible as arguments? To avoid paradox, the set constructor cannot be applied to all pluralities sanctioned by traditional plural logic. One option is to restrict the application of this and perhaps other constructors only to appropriate pluralities. Another option is to use critical plural logic (in the sense of Florio and Linnebo 2021) rather than traditional plural logic. A third option, which we pursue in this report, is to develop the theory of construction in a stage-theoretic setting. In this appendix, we remain neutral on which option is chosen.

Our theory of constructed objects relies on two distinct inductive characterizations:

1. one induction concerns the construction of objects of kind K (Ks for short):

(a) a sufficient condition for the existence of Ks (e.g. sum formation, decomposition, etc.);

(b) a necessary condition for the existence of Ks, to the effect that all Ks can be obtained by means of the aforementioned constructors.

2. Another induction concerns identity among Ks. This comprises:

(a) sufficient conditions for Ks to be identical, in the form of the laws of identification expressing the CLAP profile of Ks;

(b) a necessary condition for Ks to be identical: their identity must follow from the mentioned sufficient conditions, so we supplement these sufficient conditions with an "extremal clause" to the effect that "that's it", these are the only grounds for identifying Ks.

This is all very constructional: both Ks and identities among them are successively constructed in accordance with certain principles. And all the Ks that exist, as well as valid identities between them, can be derived from the mentioned principles.

E.1 Induction on the construction of Ks

In the simplest case, which is our focus in this appendix, every K can be built up by means of the relevant summation operator Σ and only Ks can be input to this operator. We express this by means of the following induction principle.

Induction on Construction of Ks (IC(K))

Assume

(i) every given eligible to serve as input to Σ is φ

(ii) if each of xx is φ, then so is $\Sigma(xx)$.

Then every K is φ.

Note that clause (i) involves the predicate *being a given*, which we adopt as primitive. In the case of sets, the induction principle is just induction on $\in$, which is known to be equivalent to the axiom of Foundation.

Notice that in our stage-theoretic setting, it can be proved from the well-foundedness of the stages that mereological sums satisfy this induction scheme. Since our exposition here is independent of stage theory, we assume, rather than derive, that the induction principle holds when the Ks are sums (Ss). When $K = S$, the associated induction principle will be called IC(S).

Given that the CLAP profile of sums is the full CLAP, this assumption allows us to prove something that is both interesting in its own right and useful for our subsequent discussion. The result was anticipated in Section 9.10:

Lemma 1. *Every S can be written as a sum of givens:*

$$\forall x(Sx \rightarrow \exists gg(x = \Sigma(gg)))$$

(Here, and in what follows, we let variables such as 'gg', 'hh', etc. be implicitly restricted to givens.)

Proof. The proof proceeds by the induction principle IC(S).

(i) This property holds of every given, by Collapse.

(ii) Assume next that each member x_i of xx has the property. Then each x_i can be written as $\Sigma(gg_i)$. It follows from Leibniz's law that that $\Sigma(xx) = \Sigma(yy)$, where yy is the plurality obtained from xx by replacing each x_i with $\Sigma(gg_i)$. Finally, it also follows from Leveling that $\Sigma(yy) = \Sigma(gg)$, where gg is the plurality combining all gg_i. So Leveling allows us to capture the following, intuitive identities:

$$\Sigma(x_1, \ldots, x_i, \ldots) = \Sigma(\Sigma(gg_1), \ldots, \Sigma(gg_i), \ldots) = \Sigma(gg_1, \ldots, gg_i, \ldots)$$

Thus, our conclusion follows by IC(S). □

Remark. Eventually, we will want to allow other ways to construct mereological objects, say, by decomposition. Then clause (ii) of IC(S) will no longer be sufficient; additionally, we need to require that φ be inherited under each of the new forms of construction that we adopt (see the induction principle in Section 9.16). Call the induction principle thus revised IC(S)$^+$. Notice also that Lemma 1 turns essentially on IC(S); the modified principle IC(S)$^+$ would not suffice. So this lemma is not valid in the more general setting where mereological objects can be obtained by constructors other than Σ. Thus, the induction principle IC(S) and its consequence (Lemma 1) are specific to our present setting where we have a single constructor Σ and it takes as inputs only givens and objects constructed by Σ. However, the general idea of giving an inductive characterization of all the constructed objects in terms of the relevant constructors is fully reusable and extends readily to cases with multiple constructors: we just adopt IC(S)$^+$ rather than IC(S).

E.2 From sufficient to necessary conditions for identity

Consider some construction operator Σ. The operator has a CLAP profile. We adopt, as *sufficient conditions for identity*, each of the laws C (Collapse), L (Leveling), A (Absorption), and P (Permutation) associated with this profile. For example, if C is part of the profile, we adopt the principle $\Sigma(xx) = x$, where xx comprise just x. (As noted before, the laws A and P are implicit in our choice of pluralities as arguments for the constructor.)

Having laid down sufficient conditions for identity, our next task is to formulate *necessary conditions* for objects constructed by Σ to be identical. Our aim is to formulate an extremal clause to the effect that "that's it", i.e. that all valid identities can be obtained by plural logic and the sufficient conditions for identity from the relevant CLAP profile.

To understand the nature of the task, it is useful to observe that different choices of necessary conditions can be compatible with an adopted set of sufficient conditions. A nice example concerns sets and cardinal numbers. Clearly, the sufficient conditions associated with C and L fail for both types of object; in this sense, sets and cardinal number have the same CLAP profile. Yet different necessary conditions for identity are appropriate for the two types of object. Two sets are identical only if they are constructed from the very same objects, which corresponds to the following necessary condition:

$$\Sigma(xx) = \Sigma(yy) \rightarrow xx \approx yy \tag{1}$$

Cardinal numbers, by contrast, are subject to a far less demanding necessary condition:

$$\Sigma(xx) = \Sigma(yy) \rightarrow xx \equiv yy \tag{2}$$

where $xx \equiv yy$ abbreviates a standard formalisation of the claim that xx and yy are equinumerous.

Of course, the consequent of (1) entails that of (2): coextensive pluralities are also equinumerous. This means that (2) too is valid for sets. The problem with adopting this as our only necessary condition for the identity of sets is that it leaves the door open to far too many identifications. It permits $\Sigma(xx)$ and $\Sigma(yy)$ to be identified even if this identification isn't obtained from any of the sufficient conditions associated with sets.

Our aim, then, is to formulate a necessary condition for the identity of sums that captures the idea that the only way that $\Sigma(xx)$ and $\Sigma(yy)$ could come to be identified is via the sufficient conditions for identity associated with Σ's CLAP profile. In fact, we will consider two different ways to formulate the desired extremal clause.

A general but impractical option is simply to add as extra axioms the negation of all closed identity statements whose truth is not required by the sufficient conditions for identity. This amounts to saying that we regard an identity as true only if it is required by the sufficient conditions. An advantage of this approach is that it is uniform across all different summation operators Σ of the form we consider, and far beyond. It thus explains what all the different necessary conditions have in common, namely, that they provide extremal clauses for the associated sufficient conditions.

In many particular cases, though, a far more practical option is available. We can adopt a simpler formulation of the extremal clause and show that it is equivalent to the formulation in terms of negated identity statements. In effect, we give a more practical, finitary reaxiomatisation of the theory resulting from the impractical approach.

The case of sets provides a clear illustration of the simpler approach to the extremal clause. As noted in Section 9.9, the CLAP profile of sets is $\not{C}\not{L}$AP. So there is a single sufficient condition for the identity of sets:

$$xx \approx yy \rightarrow \Sigma(xx) = \Sigma(yy) \tag{3}$$

which follows from plural logic alone. The extremal clause associated with this CLAP profile lays down that there is no way for $\Sigma(xx)$ and $\Sigma(yy)$ to be identified other than through (3). The first option for the formulation of the extremal clause is to adopt as extra axioms the negation of all closed identity statements not required by (3). This amounts to expanding the relevant theory Δ to the theory

$$\Delta^{\neq} = \Delta \cup \{t_1 \neq t_2 : \Delta \nvdash t_1 = t_2\}$$

The second, much simpler option is to expand Δ by means of a single axiom, namely, the following principle of extensionality:

$$\Sigma(xx) = \Sigma(yy) \rightarrow xx \approx yy \tag{Ext}$$

This is the converse of (3). So we have the expanded theory

$$\Delta^{+} = \Delta \cup \{(\text{Ext})\}$$

The choice of Δ^{+} can be justified by showing it equivalent, in the appropriate setting, to $\Delta^{\neq}$; that is, Δ^{+} is a reaxiomatisation of $\Delta^{\neq}$.

The case of mereological sums is less straightforward but more interesting. In addition to the logical principle (3), we here have the sufficient conditions associated with C and L. So in this case, there isn't a single sufficient condition whose converse we can adopt. We are looking for one or more necessary conditions for identity which function as an extremal clause for the mentioned sufficient conditions. We claimed in Section 9.10 that the following hyperextensional principle (Goodman 1958) provides the desired extremal clause:

$$\Sigma(gg) = \Sigma(hh) \rightarrow gg \approx hh \tag{4}$$

That is, if one sum can be constructed from givens gg and also from givens hh, then gg are the very same objects as hh. This is pleasingly similar to the criterion of identity (Ext) for sets. Notice also that, combined with Lemma 1, (4) entails that every object is the sum of a *unique* plurality of givens.

Another pleasing aspect of (4) is that it is a natural generalization of the claim that every given is a mereological atom, with parthood defined as in (D28). Then the claim that g is an atom can be formalized as:

$$\forall hh(g = \Sigma(hh) \rightarrow g \approx hh) \tag{4^{-}}$$

where the consequent is an ad hoc shorthand for '$g \prec hh \wedge \forall h(h \prec hh \rightarrow h = g)$'.

Lemma 2. *A given is a mereological atom iff it satisfies (4^{-}).*

Proof. Suppose a given g has a part x, i.e. $x \leq g$. By definition of parthood, there are yy such that $g = \Sigma(yy)$ and $x \prec yy$. By Lemma 1, there are givens hh such that $x = \Sigma(hh)$. Using Lemma 1 again and Leveling, we have the following fact: $x \leq g$ iff there are givens ii and hh such that $g = \Sigma(ii)$, $hh \preccurlyeq ii$, and $x = \Sigma(hh)$. Using this fact, it straightforward to establish the lemma. Assume that g is a mereological atom and $g = \Sigma(hh)$. Then every member of hh must

be g, i.e. $g \approx hh$. Conversely, assume that g satisfies (4^-) and $x \leq g$. It follows from (4^-) that $g \approx ii$ and thus $g \approx hh$, which entails by Collapse that $g = x$. So g is a mereological atom. □

Thus, where (4^-) says that every *given* is a unique sum of givens, i.e. the sum of itself, (4) says that every mereological object, *whether a given or not*, is a unique sum of givens.

Our next aim is to show that the general but impractical formulation of the extremal clause is equivalent to the simpler formulation based on the hyperextensional principle (4). Before we give a precise description of these theories, however, we need to describe the intended model of our construction, which these theories are attempts to describe.

E.3 An intended model of our construction

Suppose we start with a set G of givens to which we apply a summation operator Σ. We will successively construct mereological sums and identify these in accordance with the relevant sufficient conditions for identity, which in this case are the laws C and L.

We start our construction with the givens, which we assume are not sets. Let $D_0 = G$. At the first round, we construct $\Sigma(gg)$ for any gg among the elements of G. To represent these objects, consider $D_0 \cup \wp^+(D_0)$, where $\wp^+(X)$ is defined as the set of non-empty subsets of a set X. We now start identifying objects. We represent these identifications by means of an equivalence relation $\sim_1$ on $D_0 \cup \wp^+(D_0)$.

The only law that applies at this stage of the construction is C, which tells us to identify each given g with its own singleton $\{g\}$. Thus, we let $\sim_1$ extend the identity relation by also relating $g \sim_1 \{g\}$ for each given g.

The domain available after the first round of construction, D_1, is defined as the quotient $(D_0 \cup \wp^+(D_0))/\sim_1$. Clearly, we have a natural embedding of D_0 into D_1, namely, to map x to its own equivalence class.

At the next round, we proceed in a similar way, using D_1 and $\wp^+(D_1)$. That is, we build up an equivalence relation $\sim_2$ on $D_1 \cup \wp^+(D_1)$ that represents all the identifications licensed by C and L.

At this stage the powerful law L is applicable. It tells us that, for any non-empty $X \subseteq D_1$, we have $X \sim_2 \cup X$, where $X \in D_1$. Thus, the L allows us to identify each element of $\wp^+(D_1)$ with some element of D_1. In this sense, applying the summation operator a second time produced no new objects: every object thus produced is identified with an object already in D_1. We conclude that the intended model of our construction of mereological sums, where we make every identification licensed by C and L, *but no further identifications*, is isomorphic to D_1, which in turn is isomorphic to $\wp^+(G)$.

E.4 Equivalent formulations of the extremal clause

We want to state precisely the equivalence between our two formulations of the extremal clause. We are studying pluralities that may be infinite in size and that contain objects on which we have not yet characterized identity. Then there are operations on these pluralities and associated sufficient conditions for the identification of the outputs of the operations. The extremal clause expresses that no two objects are identical unless they are identified as a result of these sufficient conditions. We would like a framework that provides a very direct way of stating extremal clauses. So we adopt one in which terms have the same infinitary structure as the objects we are studying. Crucially, instead of a plurality xx, we will use a constant t_x for each $x \prec xx$, and the conjunction $\&_{t_x}$ of all the t_x.

We start with first-order logic, which we expand by accommodating the following items in a simultaneous recursion (where the plural terms and the singular terms are defined simultaneously):

(a) a term conjunction & that applies to any set of singular terms, possibly infinite, and yields a plural term (for convenience, we will write $\&_i t_i$ for $\&\{t_i : i \in I\}$, where I is some relevant, non-empty index set);

(b) the binary predicate $\prec$ for plural membership, applying to singular term and a plural one;

(c) a constructor Σ that applies to any plural term and yields a singular term;

(d) a disjunction $\bigvee$ that applies to any set of formulas, possibly infinite, and yields a formula; intuitively $\bigvee\{\varphi_n : 0 \leq n \leq 5\}$ represents $\varphi_0 \vee \ldots \vee \varphi_5$ (here too we simply the notation by writing $\bigvee_i \varphi_i$ for $\bigvee\{\varphi_i : i \in I\}$, where I is some relevant, non-empty index set);

(e) a unique canonical constant g_α for each given, with α from an appropriate index set A;

(f) a special predicate G for *being a given*.

So we have an infinitary expansion of first-order logic whose structure is similar to that of plural logic. Terms of the form $\&_i t_i$ are here the only plural terms.

Within this framework, we characterise the theory Γ that will be the basis for the two formulations of the extremal clause. We want to be able to carry out infinitary reasoning with disjunctions, and we define infinitary conjunctions via versions of De Morgan's laws. So we assume an infinitary proof system containing axiom schemes and rules of inference supporting the required reasoning (see, for example, Dickmann 1975, Appendix C).

Plural terms, i.e. terms of the form $\&_i t_i$, are governed by axioms that specify the members of the denoted pluralities:

$$\forall x(x \prec \&_i t_i \;\leftrightarrow\; \bigvee_i x = t_i) \tag{5}$$

Plural inclusion and plural identity are defined, respectively, as:

$$\&_i t_i \preccurlyeq \&_j t_j \;\leftrightarrow\; \forall x(x \prec \&_i t_i \rightarrow x \prec \&_j t_j) \tag{6}$$

$$\&_i t_i \approx \&_j t_j \leftrightarrow\; \&_i t_i \preccurlyeq \&_j t_j \wedge \&_i t_i \preccurlyeq \&_j t_j \tag{7}$$

Plural terms are governed by Leibniz's law:

$$\&_i t_i \approx \&_j t_j \rightarrow\; (\varphi(\&_i t_i) \leftrightarrow \varphi(\&_j t_j)) \tag{8}$$

The status of the givens is captured by the axiom:

$$\forall x(x \prec \&_\alpha g_\alpha \;\leftrightarrow\; Gx) \tag{9}$$

where $\&_\alpha g_\alpha$ abbreviates $\&\{g_\alpha : \alpha \in A\}$, the term conjunction of all constants for givens. We also assume that Γ proves the appropriate distinctness claims involving givens. So, for every $i, j \in A$ with $i \neq j$, we add the axiom:

$$g_i \neq g_j \tag{10}$$

Note that, as a result, Γ decides coextensionality statements between pluralities of givens (i.e. statements of the form $\&_\beta g_\beta \approx \&_\delta g_\delta$).

Next, the constructor Σ satisfies C:

$$\forall x \forall y(x \prec \&_i t_i \wedge y \prec \&_i t_i \rightarrow x = y) \rightarrow \forall x(x \prec \&_i t_i \rightarrow \Sigma(\&_i t_i) = x) \tag{11}$$

(In words: if $\&_i t_i$ includes a single object, then its sum is identical with that object.)

Finally, the constructor satisfies L:

$$\begin{aligned} &\forall x(x \prec \&_i t_i \rightarrow (x \prec \&_j t_j \vee \bigvee_K x = \Sigma(\&_k t_k))) \wedge \\ &\forall y(y \prec \&_j t_j \rightarrow (y \prec \&_i t_i \vee \bigvee_K (y \prec \&_k t_k \wedge \Sigma(\&_k t_k) \prec \&_i t_i))) \;\rightarrow \\ &\qquad\qquad \Sigma(\&_i t_i) = \Sigma(\&_j t_j) \end{aligned} \tag{12}$$

where K ranges over all non-empty subsets of J, the index set of the t_j, and for each K, k ranges over all elements of K. The two disjunctions that range over K simulate existential quantification over the supluralities of $\&_j t_j$. (In words: two pluralities, one obtained from the other by replacing some pluralities of objects with their respective sums, yield the same sum.)

We contend that the theory Γ is consistent. This can be proved by constructing a set-theoretic model based on the constants $\{g_\alpha : \alpha \in A\}$, with arbitrary sums, and where we identify in accordance with C and L. We now observe that Γ is satisfied in the model described in Section E.3 and is therefore consistent.

We also contend that Γ proves all the identity statements involving two terms that follow from our sufficient conditions for identification. For, whenever the antecedent of one of these conditions obtains, then Γ proves that it does, and

thus it proves the ensuing identity statement. The only hard case is that of Leveling. Suppose that the antecedent of an instance of Leveling holds. To see that Γ proves this antecedent, we need only observe that Γ handles the relevant disjunctions in truth-functional logic. The upshot is that when the antecedent of L holds, then Γ proves that it holds. This ensures that Γ can utilise the fact that it contains L to prove the desired identity.

Our next result shows that Γ proves, of every closed singular term, that it can be rewritten in a certain canonical form, namely, as a sum of givens.

Proposition 1. *For every closed singular term t, there is a set $\{g_\beta : \beta \in B\}$, with $B \subseteq A$, such that $\Gamma \vdash t = \Sigma(\&_\beta g_\beta)$.*

Proof. The proof proceeds by induction on the construction of t. There are just two options: t could be a term g_α for a given, or t could be $\Sigma(\&_i t_i)$ for some terms t_i already constructed. Suppose t is a constant for a given, say, g. Then, by C, we have $\Gamma \vdash g = \Sigma(\&\{g\})$. Next, suppose that t is $\Sigma(\&_i t_i)$. By inductive hypothesis, we have that, for every i, $\Gamma \vdash t_i = \Sigma(\&C_i)$ for some $C_i \subseteq \{g_\alpha : \alpha \in A\}$. Let $C = \cup_i C_i$. We want to show that $\Gamma \vdash t = \Sigma(\&C)$, i.e. $\Gamma \vdash \Sigma(\&_i t_i) = \Sigma(\&C)$. Since $\Gamma \vdash t_i = \Sigma(\&C_i)$ for every i, $\Gamma \vdash \&_i t_i \approx \&_i \Sigma(\&C_i)$. By Leibniz's law, we thus obtain $\Gamma \vdash \Sigma(\&_i t_i) = \Sigma(\&_i \Sigma(\&C_i))$. It remains only to apply L to simplify the right-hand term of this last identity. To do so, we observe that it is provable in Γ that $\&_i \Sigma(\&C_i)$ is obtained from $\&C$ by replacing every subset C_i of C with its sum $\Sigma(\&C_i)$. We can then apply L and obtain $\Gamma \vdash \Sigma(\&_i \Sigma(\&C_i)) = \Sigma(\&C)$. By transitivity of identity, we thus obtain $\Gamma \vdash \Sigma(\&_i t_i) = \Sigma(\&C)$, as desired. Since these are the only ways to construct singular terms, this exhausts all the cases. □

We observe that Proposition 1 is a syntactic analogue of the earlier Lemma 1. Where the proof of the lemma uses plural quantification and the induction principle IC(S), the proof of the proposition uses infinitary logic and induction on the construction of terms.

As we have seen, the general but impractical way to state the extremal clause is to adopt the negation of every identification that doesn't follow from the sufficient conditions:

$$\Gamma^{\neq} = \Gamma \cup \{t_1 \neq t_2 : \Gamma \nvdash t_1 = t_2\}$$

Naturally, this expresses that we identify no more than what is required by the sufficient conditions. Since Γ is consistent, so is $\Gamma^{\neq}$. This theory is consistent, as it is satisfied in the model mentioned above.

Recall that the more practical reaxiomatisation of the resulting theory is based on the hyperextensional principle that if one sum can be constructed from givens gg and also from givens hh, then gg are the very same objects as hh. In the present setting, this can be expressed as follows:

$$\&_\beta g_\beta \preccurlyeq \&_\alpha g_\alpha \ \wedge\ \&_\delta g_\delta \preccurlyeq \&_\alpha g_\alpha \wedge\ \Sigma(\&_\beta g_\beta) = \Sigma(\&_\delta g_\delta) \ \rightarrow\ \&_\beta g_\beta \approx \&\delta g_\delta \qquad \text{(HEP)}$$

where β and γ range over subsets of the index set A. We obtain the desired reaxiomatisation by adding (HEP) to the base theory Γ:

$$\Gamma^+ = \Gamma \cup \{(\text{HEP})\}$$

This theory too is satisfied in the model mentioned above. So it is consistent.

Lemma 3. *Both $\Gamma^{\neq}$ and Γ^+ decide all identity statements flanked by closed singular terms.*

Proof. For $\Gamma^{\neq}$, the claim is trivial: either Γ proves the identity $t_1 = t_2$ or else its negation is added to $\Gamma^{\neq}$. Next, consider Γ^+ and the identity $t_1 = t_2$. By Proposition 1, there are constants for givens $\{g_\beta : \beta \in B\}$ and $\{g_\delta : \delta \in D\}$, with $B, D \subseteq A$, such that Γ proves $t_1 = \Sigma(\&_\beta g_\beta)$ and $t_2 = \Sigma(\&_\delta g_\delta)$. Furthermore, Γ^+ decides the coextensionality statement $\&_\beta g_\beta \approx \&_\delta g_\delta$, since Γ does. There are two options. Suppose Γ^+ proves the coextensionality. Then by Leibniz's law, it also proves the identity $t_1 = t_2$. Alternatively, if Γ^+ *dis*proves the coextensionality, it also disproves the mentioned identity by (HEP). □

We are now ready to prove our claim that $\Gamma^{\neq}$ and Γ^+ are equivalent and thus the hyperextensionality principle does the job. This justifies the use of the hyperextensionality principle as our necessary condition.

Theorem 1. *(a)* $\Gamma^{\neq} \vdash t_1 = t_2 \;\Leftrightarrow\; \Gamma^+ \vdash t_1 = t_2$*;*

(b) $\Gamma^{\neq} \vdash t_1 \neq t_2 \;\Leftrightarrow\; \Gamma^+ \vdash t_1 \neq t_2$.

Proof. (a) Suppose $\Gamma^{\neq} \vdash t_1 = t_2$. Suppose, for contradiction, that Γ does not prove $t_1 = t_2$. Then $\Gamma^{\neq} \vdash t_1 \neq t_2$, which contradicts our observation that $\Gamma^{\neq}$ is consistent. Hence Γ, and therefore also Γ^+, proves $t_1 = t_2$, as desired.

Suppose $\Gamma^+ \vdash t_1 = t_2$. By Proposition 1, there are constants for givens $\{g_\beta : \beta \in B\}$ and $\{g_\delta : \delta \in D\}$, with $B, D \subseteq A$, such that Γ proves $t_1 = \Sigma(\&_\beta g_\beta)$ and $t_2 = \Sigma(\&_\delta g_\delta)$. So $\Gamma^+ \vdash \Sigma(\&_\beta g_\beta) = \Sigma(\&_\delta g_\delta)$. Hence, by the hyperextensional principle (HEP), $\Gamma^+ \vdash \&_\alpha g_\alpha \approx \&_\beta g_\beta$. Since Γ decides coextensionality statements between pluralities of givens, and since Γ^+ is consistent, it follows that $\Gamma \vdash \&_\beta g_\beta \approx \&_\delta g_\delta$. By Leibniz's law, $\Gamma \vdash \Sigma(\&_\beta g_\beta) = \Sigma(\&_\delta g_\delta)$, hence $\Gamma^{\neq}$ proves this as well, which is what we wanted to show.

Claim (b) follows immediately from (a), given that both theories are consistent and decide closed identity statements (by Lemma 3). □

Our equivalence result concerning the two ways of formulating the extremal clause makes crucial use of an infinitary language. What, one may wonder, does this tell us about the context of an ordinary finitary language, i.e. a language without the infinitary resources described above? We have already observed that the hyperextensionality principle (HEP) is sound with respect to the intended model. We additionally contend that (HEP) provides a complete basis for distinguishing mereological sums. Consider any closed singular terms t_1 and t_2. Suppose their referents are distinguished in the intended model (and thus also $\Gamma^{\neq} \vdash t_1 \neq t_2$). Suppose a theory F in a some finitary language includes

(4), i.e. our earlier finitary statement of the hyperextensionality principle, and yet fails to prove this distinctness claim. Then this incompleteness does not arise because (4) is an inadequate basis for distinguishing sums, it arises because of the weakness of F. That is, F fails to prove *either* that the two terms in question can be rewritten in canonical form as sums of givens *or* that the associated coextensionality statement involving pluralities of givens is false. In a nutshell, F "knows" all there is to know about how to distinguish sums but lacks the information needed to apply this knowledge.

E.5 Further investigations

The above discussion calls for further investigations, which we plan to undertake in future work. In particular, we would like to:

1. document the reusability of our framework by showing that it is possible to lift the assumption that Σ is the only constructor that yields mereological sums. For example, one may use a deconstructor that partitions a mereological sum into proper parts.

2. extend our account to constructors with other CLAP profiles, for example a constructor satisfying Collapse but not Leveling. This combination would give Quinean sets where every individual is identified with its own singleton.

3. investigate the use of several constructors side by side, especially in contexts where one can identify objects constructed from different constructors. For example, one may identify Quinean sets of cardinality two or more with the corresponding ordinary sets. This means adopting further sufficient conditions for identity. Again, one can formulate and examine appropriate extremal clauses, which allow us to distinguish objects not identified by the new sufficient conditions.

4. develop a more systematic framework for studying the equivalence of alternative formulations of extremal clauses.

F Proof of consistency of the Core Constructional Theory

We prove consistency by constructing a model for the CCT, in this case a set-theoretic model. For convenience, we will work in set theory with urelements. Since our urelements form a set, it is routine to translate this impure set theory into a corresponding pure set theory. Our metatheory will mostly be ZFC, but at times it will also be useful to invoke Morse-Kelley set theory, which adds to ZFC a single layer of classes on top of all the sets.

The model we construct validates the full CCT, including the optional additional principle (*) to the effect that every successor stage is a maximal extension

(Section 9.15). In future work, it would be interesting to consider alternative, or more general, set-theoretic constructions which model other stage-theoretic axioms and assumptions.

As a basis for the model, we assume *two disjoint sorts of urelements*:

- A set I of index objects that encode the basic type of the objects that are modeled: **s** for set, **m** for (mereological) individual, **l** for left object, **r** for right object, **p** for pair, and **t** for trace. (We do not need to include an index object for the derived type union.) That is, $I = \{\mathbf{s}, \mathbf{m}, \mathbf{l}, \mathbf{r}, \mathbf{p}, \mathbf{t}\}$. We require that the index objects are not used in any other way in the set-theoretic model.

- A set G of givens.

We also assume G has the following subset:

- A set W of the six constants indicating types of construction. So $W = \{c_{\text{set}}, c_{\text{sum}}, c_{\text{left}}, c_{\text{right}}, c_{\text{pair}}, c_{\text{union}}\}$.

Other than the stages, which will be treated separately, we model each object as a pair of an index object, which specifies what *kind* of object is being represented, and a set, which specifies the particular objects of this kind that is being represented. In characterizing the set-theoretic model, we adopt the usual notation for ordered tuples. The notation (a, b) stands for the Kuratowski pair, i.e. $(a, b) =_{\text{def}} \{\{a\}, \{a, b\}\}$. The notation (a, b, c) stands for $((a, b), c)$. And so on for longer tuples.

A remark about the role of the index objects: their function is to avoid clashes between types, or more precisely, to prevent confusion over which type of object is represented by a certain set, which would also lead to unclarity about *which object* is represented by the set in question. An example: without the index objects, the set $\{\{a\}, \{a, b\}\}$ might be taken to represent either itself, i.e. a set of rank 2, or the pair of a and b.

Here is the idea. Consider an object z:

- z represents a *set* x iff $z = (\mathbf{s}, x)$.

- z represents the mereological sum of a set x of givens iff there is an $x \subseteq G$ such that $z = (\mathbf{m}, x)$. We say that z represents an *individual* iff z represents the mereological sum of a set of givens.

- z represents a *left object* x iff $z = (\mathbf{l}, x)$.

- z represents a *right object* x iff $z = (\mathbf{r}, x)$.

- z represents a *pair* x iff $z = (\mathbf{p}, x)$.

- z represents a *trace* iff $z = (\mathbf{t}, x_1, x_2, x_3)$ and one of the following is the case:

  (i) $x_3 = c_{\text{set}}$, x_1 represents a set y, and x_2 is a member of y;

(ii) $x_3 = c_{\text{sum}}$, x_1 represents a mereological sum of a set y of givens, and x_2 represents a sum of a subset of y;

(iii) $x_3 = c_{\text{left}}$, x_1 represents a left object y, and x_2 is a member of y;

(iv) $x_3 = c_{\text{right}}$, x_1 represents a right object y, and x_2 is a member of y;

(v) $x_1 = c_{\text{pair}}$, x_1 represents a pair y, and x_2 is a member of y;

(vi) $x_1 = c_{\text{union}}$, x_1 represents a set y, and x_2 represents a set $y' \subseteq y$.

We now specify the model. We define a construction operation C such that $C(X)$ will model the result of undertaking all of our forms of construction on the basis of the objects represented by elements of X. The operation will also model the introduction of the appropriate traces based on these constructions. That is, given a set X, we define $C(X)$ as the set whose elements are all and only the following objects:

- for any $x \subseteq X$ such that every element of x represents a settable object, $(\mathbf{s}, x) \in C(X)$ and $(\mathbf{t}, (\mathbf{s}, x), y, c_{\text{set}}) \in C(X)$ for every $y \in x$;
- for any $x \subseteq X$ such that every member of x is summable, $(\mathbf{m}, z) \in C(X)$, where z is the union of all of the sets that figure as second coordinates of elements of x, and $(\mathbf{t}, (\mathbf{m}, z), y, c_{\text{sum}}) \in C(X)$ for every $y \in x$;
- for any singleton $x \subseteq X$ whose element y represents a positionable object, $(\mathbf{l}, x) \in C(X)$ and $(\mathbf{t}, (\mathbf{l}, x), y, c_{\text{left}}) \in C(X)$;
- for any singleton $x \subseteq X$ whose element y represents a positionable object, $(\mathbf{r}, x) \in C(X)$ and $(\mathbf{t}, (\mathbf{r}, x), y, c_{\text{right}}) \in C(X)$;
- for any $x \subseteq X$ with exactly two elements y_1 and y_2 representing a left object and a right object, $(\mathbf{p}, x) \in C(X)$, $(\mathbf{t}, (\mathbf{p}, x), y_1, c_{\text{pair}}) \in C(X)$, and $(\mathbf{t}, (\mathbf{p}, x), y_2, c_{\text{pair}}) \in C(X)$;
- for any $x \subseteq X$ such that every element of x represents a set, $(\mathbf{t}, (\mathbf{s}, z), y, c_{\text{union}}) \in C(X)$ for every $y \in x$, where z is the union of all of the sets that figure as second coordinates of elements of x.

Let G^* be $\{(\mathbf{m}, \{g\}) : g \in G\}$.

By set-theoretic recursion, we now define a sequence M_α such that

- $M_0 = G^*$;
- $M_{\alpha+1} = M_\alpha \cup C(M_\alpha)$;
- $M_\lambda = \bigcup_{\gamma<\lambda} M_\gamma$.

As usual (see Kunen 1980), we can talk about M as the union of all of the M_α, exactly as we talk about the set-theoretic universe V as the union of all of the ranks V_α. In fact, for the purposes of interpreting traditional plural logic, it is convenient to use as our metatheory Morse-Kelley rather than ZFC. Then M can be a proper class. Alternatively, if we use ZFC + "there is an inaccessible cardinal", M can be a subset of M_κ for κ the first inaccessible.

Lemma 1. *When* $(\mathbf{m}, x) \in M$*, then* $x \subseteq G$*.*

Proof. A straightforward induction on the construction of M. □

Now, M is almost the model we need. To represent the stages, which do not emerge from constructions, we add to M appropriate ordinals from our metatheory. We assume that there is no overlap between M and these ordinals. Let M^* be the resulting model. If we use Morse-Kelley, M^* is proper class: the union of M and the proper class of all ordinals. If we use ZFC + "there is an inaccessible cardinal", M^* is a set: the union of M and the set of ordinals less than the first inaccessible.

We interpret the CCT in M^*. Plural variables are interpreted as ranging over non-empty subclasses of M^* (if we use Morse-Kelley class theory) or non-empty subsets of M^* (if we use ZFC + "there is an inaccessible cardinal"). Plural membership is interpreted as set- or class-theoretic membership, i.e. '$\prec$' is true of a, b iff $a \in b$. The interpretation of the basic vocabulary of the CCT is given by the following clauses:

- '$\textsc{Construct}(z : xx, y)$' is true of a, b, c iff $c \in W$ and:
  - if $c = c_{\text{set}}$, $a = (\mathbf{s}, b)$;
  - if $c = c_{\text{sum}}$, $a = (\mathbf{m}, d)$ and d is the union of all of the sets that figure as second coordinates of elements of b;
  - if $c = c_{\text{left}}$, $a = (\mathbf{l}, b)$;
  - if $c = c_{\text{right}}$, $a = (\mathbf{r}, b)$;
  - if $c = c_{\text{pair}}$, $a = (\mathbf{p}, b)$;
  - if $c = c_{\text{union}}$, then every element of b represents a set and $a = (\mathbf{s}, d)$, where d is the union of all of the sets that figure as second coordinates of elements of b.
- '$\textsc{Individual}$' is true of a iff $a = (\mathbf{m}, x)$, for some x.
- '$\textsc{HasOutput}(w, z)$' is true of a, b iff $a = (\mathbf{t}, b, x_2, x_3)$, for some x_2, x_3.
- '$\textsc{HasInput}(w, x)$' is true of a, b iff $a = (\mathbf{t}, x_1, b, x_3)$, for some x_1, x_3.
- '$\textsc{HasType}(w, y)$' is true of a, b iff $a = (\mathbf{t}, x_1, x_2, b)$, for some x_1, x_2.
- '$\textsc{Stage}$' is true of all and only the ordinals in M^*, where '$\trianglelefteq$' is true of α and β iff $\alpha \leq \beta$, and '$x@s$' is true of a and α iff $a \in M_\alpha$.

The language of the CCT also contains six special constants, which have been included among the givens. So we let these constant interpret themselves. Finally, the interpretation of the vocabulary introduced by definition is determined by the interpretation of the defining expression.

Theorem 1. *All of the axioms of the CCT are satisfied in this model.*

Proof. We sketch the reasoning why the axioms satisfied in the model, proceeding by groups of axioms.

- The axioms of plural logic are satisfied because plural variables are interpreted as ranging over non-empty subcollections of M^*. These subcollections will be either subclasses or subsets, depending on whether we use MK or ZFC + "there is an inaccessible". (Notice that to validate comprehension axioms that give pluralities not bounded by a stage, we need to go beyond ZFC. This is the only place where this is needed.)

- The axioms concerning stages are satisfied because these axioms are true of the canonical ordering ($\alpha \leq \beta$) of the ordinals. Notice that the proofs that the stage-theoretic axioms of Infinity and Replacement are satisfied uses the corresponding axioms of set theory.

- The axioms concerning the initial stage are satisfied because M_0 has been chosen so as to be non-empty and to contain the interpretations of the six special constants.

- The axioms concerning what exists at a stage are satisfied: every non-stage exists at a stage; the stages are cumulative; limit stages just accumulate the preceding stages; stages with identical domains are identical; and successor stages contain only the domain of the preceding stage, objects that can be constructed from this domain, and appropriate traces.

- There are two axioms concerning the generic constructor. The definition of the construction operator C and the interpretation of the predicate 'CONSTRUCT' ensures that objects of the same type, constructed from the same pluralities, are the same. Moreover, the interpretation has been constrained so that if an object is constructed from a plurality using a parameter, the parameter is a type of construction. So both axioms are satisfied.

- The classification axioms are satisfied by the way in which different types of objects have been encoded in the model. This encoding represents different types of constructed objects by means of different indices, except for unions, which are sets.

- The axioms concerning the set constructor are satisfied. The definition of the construction operator C implies that every plurality of settable objects existing at a stage (and no other plurality at that stage) forms a set at the next. Moreover, since the stages are well-founded, the elements of a

set exist prior to the set. The extensionality of sets in the metalanguage secures the extensionality of sets in the object language, which means that the injectivity axiom is satisfied. The set-theoretic version of the axiom that there is an infinite stage holds at M_ω and is thus satisfied.

- The axioms concerning the sum constructor are satisfied because of the definition of the construction operator C and of the fact that the sums are tracked in our model by the unique set of givens from which they are constructed, which also implies that Collapse and Leveling are satisfied.

- The axioms concerning the left and right constructors are satisfied. The definition of the construction operator C implies that every singleton of a positionable object existing at a stage forms a position object at the next. Since the stages are well-founded, the element of a singleton of a positionable object exists prior to the resulting position object. Given our representation of position objects in the model, the extensionality of sets in the metalanguage secures the extensionality of position objects in the object language. This means that the injectivity axioms are satisfied.

- The axioms concerning the pair constructor are satisfied. The definition of the construction operator C implies that every set of exactly one left object and one right object existing at a stage forms a pair a the next. Since the stages are well-founded, the position objects exist prior to the pair. Given our representation of pairs in the model, the extensionality of sets in the metalanguage secures the extensionality of pairs in the object language. So the injectivity axiom is satisfied.

- Unlike other constructors, the union constructor is derived. Applications of this constructor yield sets rather than *sui generis* objects. The interpretation of the predicate for the generic constructor ensures that, when the construction type is union, the result is the appropriate set. So the single axiom concerning this constructor is satisfied.

- The axioms concerning traces are satisfied because the definition of the construction operator C ensures that all required traces are generated at the appropriate stages. Together with each constructed object, C adds traces with the appropriate information about input, output, and type of the construction.

- Finally, the model has been constructed so as to satisfy the axiom that every successor stage is maximal. This is because the construction operation C yields all of our forms of construction from the objects at a given stage, and it generates all required traces.

□

G Axioms

This appendix collects the axioms included in the body of the report, providing a single point of reference for all the axioms of the CCT. As part of the process of conforming to [ISO/IEC 21838-1:2021 Information technology — Top-level ontologies (TLO)] and building the Foundation Data Model, this appendix will be used (in a subsequent project) as the master input to a computerised translation into the Common Logic Interchange Format (CLIF) as defined in ISO/IEC 24707:2018 – Annex A. [ISO/IEC 24707:2018 Information technology — Common Logic (CL) — A framework for a family of logic-based languages]. To ensure data quality, we are currently developing an automated process to extract the axioms from the body of the report into this master appendix, and to convert these axioms into CLIF.

G.1 Primary axioms

(A1) $\forall xx \exists y\ y \prec xx$

(Every plurality has at least one member.)

(A2) $\forall xx \forall yy(\forall z(z \prec xx \ \leftrightarrow z \prec yy) \ \rightarrow (\varphi(xx) \ \leftrightarrow \ \varphi(yy)))$

(Coextensive pluralities satisfy the same formulas.)

(A3) $\exists x \varphi(x) \rightarrow \exists xx \forall x(x \prec xx \ \leftrightarrow \ \varphi(x))$

(If something is φ, then there are some things that are all and only the φs. That is, if something is φ, then the φs exist.)

(D1) $xx \preccurlyeq yy \ \leftrightarrow \ \forall z(z \prec xx \rightarrow z \prec yy)$

(Some things xx are among yy when everything that is one of xx is one of yy.)

(D2) $xx \approx yy \ \leftrightarrow \ (xx \preccurlyeq yy \ \wedge \ yy \preccurlyeq xx)$

(Two pluralities xx and yy are identical if and only if xx are among yy and yy are among xx, i.e. xx and yy have the same members.)

(A4) $\forall s\ s \trianglelefteq s$

($\trianglelefteq$ is reflexive.)

(A5) $\forall s \forall t(s \trianglelefteq t \ \wedge \ t \trianglelefteq s \ \rightarrow s = t)$

($\trianglelefteq$ is antisymmetric.)

(A6) $\forall s_0 \forall s_1 \forall s_2(s_0 \trianglelefteq s_1 \ \wedge \ s_1 \trianglelefteq s_2 \ \rightarrow s_0 \trianglelefteq s_2)$

($\trianglelefteq$ is transitive.)

(A7) $\forall s \forall t(s \trianglelefteq t \ \vee \ t \trianglelefteq s)$

($\trianglelefteq$ is connected.)

(D3) $s \lhd t \leftrightarrow (s \trianglelefteq t \wedge s \neq t)$

(A8) $\forall ss \exists s (s \prec ss \wedge \neg \exists t (t \prec ss \wedge t \lhd s))$

(The stages are well-founded by $\trianglelefteq$. That is, for any plurality ss of stages, there is a member s that is first among ss with respect to the order $\trianglelefteq$.)

(A9) $\forall s \exists t \; s \lhd t$

(For every stage, there is a strictly later stage.)

(D4) $\textsc{Succ}(s, t) \leftrightarrow s \lhd t \wedge \neg \exists u \; (s \lhd u \wedge u \lhd t)$

(A10) $\exists t (\exists s \; s \lhd t \wedge \forall s (s \lhd t \rightarrow \exists u (s \lhd u \wedge u \lhd t)))$

(There is a stage that is after some stage and is not immediately after any other stage.)

(D5) $xx@@s \;\leftrightarrow\; \forall x (x \prec xx \rightarrow x@s)$

(A11)

$$xx@@s \wedge \forall x (x \prec xx \rightarrow \exists y (\neg \textsc{Stage}(y) \wedge \forall z (\psi(x, z) \leftrightarrow y = z))) \;\rightarrow\; \exists t \forall x (x \prec xx \rightarrow \forall y (\psi(x, y) \rightarrow y@t))$$

(Suppose that xx exist at s and that $\psi(x, y)$ represents a function mapping objects among xx to objects other than stages. Then there is a stage t such that what exists at t includes the image under ψ of every member of xx.)

(D6) $\textsc{Init}(s) \leftrightarrow \neg \exists t \; t \lhd s$

(D7) $\textsc{Given}(x) \leftrightarrow \exists s (\textsc{Init}(s) \wedge x@s)$

(A12) $\exists x \; \textsc{Given}(x)$

(There is some given.)

(A13) $\textsc{Given}(c_{\text{set}}) \;\wedge\; \textsc{Given}(c_{\text{sum}}) \;\wedge\; \textsc{Given}(c_{\text{left}}) \;\wedge\; \textsc{Given}(c_{\text{right}}) \;\wedge\; \textsc{Given}(c_{\text{pair}}) \;\wedge\; \textsc{Given}(c_{\text{union}})$

(The givens include: c_{set}, c_{sum}, c_{left} c_{right}, c_{pair}, c_{union}.)

(A14) $c_{\text{set}} \neq c_{\text{sum}} \;\wedge\; c_{\text{set}} \neq c_{\text{left}} \;\wedge\; c_{\text{set}} \neq c_{\text{right}} \;\wedge\; c_{\text{set}} \neq c_{\text{pair}} \;\wedge\; c_{\text{set}} \neq c_{\text{union}} \;\wedge\; c_{\text{sum}} \neq c_{\text{left}} \;\wedge\; c_{\text{sum}} \neq c_{\text{right}} \;\wedge\; c_{\text{sum}} \neq c_{\text{pair}} \;\wedge\; c_{\text{sum}} \neq c_{\text{union}} \;\wedge\; c_{\text{left}} \neq c_{\text{right}} \;\wedge\; c_{\text{left}} \neq c_{\text{pair}} \;\wedge\; c_{\text{left}} \neq c_{\text{union}} \;\wedge\; c_{\text{right}} \neq c_{\text{pair}} \;\wedge\; c_{\text{right}} \neq c_{\text{union}} \;\wedge\; c_{\text{pair}} \neq c_{\text{union}}$

(D8)

$$\textsc{Type}(y) \;\leftrightarrow (y = c_{\text{set}} \;\vee\; y = c_{\text{sum}} \;\vee\; y = c_{\text{left}} \vee y = c_{\text{right}} \;\vee\; y = c_{\text{pair}} \;\vee\; y = c_{\text{union}})$$

(A15) $\forall x(\neg \textsc{Stage}(x) \rightarrow \exists s\, x@s)$

(Everything that isn't a stage exists at some stage.)

(A16) $\forall x(x@s \leftrightarrow x@t) \rightarrow s = t$

(Stages with identical domains are identical.)

(A10) $s \trianglelefteq t \wedge x@s \rightarrow x@t$

(When one stage precedes another, then everything that exists at the former also exists at the latter.)

(D9) $\text{LUB}(t, ss) \leftrightarrow \forall s(s \prec ss \rightarrow s \trianglelefteq t) \wedge \forall t'(\forall s(s \prec ss \rightarrow s \trianglelefteq t') \rightarrow t \trianglelefteq t')$

(According to the definition, t is the least upper bound of ss if and only if two conditions are met: (i) t is an upper bound of ss, i.e. t is after, or equal to, any stage in ss; (ii) t is the least among the upper bounds of ss, i.e. t is before, or equal to, any upper bound of ss.)

(A17) $\text{LUB}(t, ss) \rightarrow \forall x(x@t \rightarrow \exists s(s \prec ss \wedge x@s))$

(Suppose t is the least upper bound of some stages ss. Then everything that exists at t exists at some of ss.)

(D10) $\textsc{ConstrFrom}(x, s) \leftrightarrow \exists xx \exists y(xx@@s \wedge \textsc{Type}(y) \wedge \textsc{Construct}(x : xx, y))$

(A18) $\textsc{Succ}(s, t) \wedge x@t \;\rightarrow\; x@s \vee \textsc{ConstrFrom}(x, s) \vee \textsc{Trace}(x)$

(Everything that exists at a successor stage either existed at the predecessor stage, or is constructed from something at that stage, or is a trace.)

(D11) $\textsc{Individual}(x) \leftrightarrow (\textsc{Given}(x) \;\vee\; \exists xx\; \textsc{Construct}(x : xx, c_{\text{sum}}))$

(Any object is an individual if and only if it is either a given or a sum.)

(A19)

$$\textsc{Construct}(z_1 : xx_1, y_1) \;\wedge\; \textsc{Construct}(z_2 : xx_2, y_2) \wedge y_1 = y_2 \;\wedge\; xx_1 \approx xx_2 \;\rightarrow\; z_1 = z_2$$

(Any two objects of the same type, constructed from the same pluralities, are the same.)

(A20) $\textsc{Construct}(z : xx, y) \;\rightarrow\; \textsc{Type}(y)$

(If an object z is constructed from xx using y, then y is a type.)

(D12) $\textsc{Set}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{set}})$

(D13) $\textsc{Sum}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{sum}})$

(D14) $\textsc{Left}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{left}})$

(D15) $\textsc{Right}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{right}})$

(D16) $\textsc{Pair}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{pair}})$

(D17) $\textsc{Union}(x : xx) \leftrightarrow \textsc{Construct}(x : xx, c_{\text{union}})$

(D18) $\textsc{IsSet}(x) \leftrightarrow \exists xx \textsc{Set}(x : xx)$

(D19) $\textsc{IsSum}(x) \leftrightarrow \exists xx \textsc{Sum}(x : xx)$

(D20) $\textsc{IsLeft}(x) \leftrightarrow \exists xx \textsc{Left}(x : xx)$

(D21) $\textsc{IsRight}(x) \leftrightarrow \exists xx \textsc{Right}(x : xx)$

(D22) $\textsc{IsPair}(x) \leftrightarrow \exists xx \textsc{Pair}(x : xx)$

(D23) $\textsc{IsUnion}(x) \leftrightarrow \exists xx \textsc{Union}(x : xx)$

(D24)

$$\begin{aligned}&\textsc{Injective}(y) \;\leftrightarrow\; \textsc{Type}(y) \wedge \\ &\quad \forall z_1 \forall xx_1 \forall z_2 \forall xx_2 (\textsc{Construct}(z_1 : xx_1, y) \;\wedge\; \textsc{Construct}(z_2 : xx_2, y) \;\rightarrow \\ &\qquad\qquad (z_1 = z_2 \;\rightarrow\; xx_1 \approx xx_2))\end{aligned}$$

(A21)

$$\begin{aligned}&\forall z_1 \forall z_2 \forall xx_1 \forall xx_2 \forall y_1 \forall y_2 (\textsc{Construct}(z_1 : xx_1, y_1) \wedge \textsc{Construct}(z_2 : xx_2, y_2) \\ &\qquad\qquad \wedge y_1 \neq y_2 \wedge y_1 \neq c_{\text{union}} \wedge y_2 \neq c_{\text{union}} \rightarrow z_1 \neq z_2) \wedge \\ &\forall z (\exists xx \exists y \textsc{Construct}(z : xx, y) \rightarrow \neg \textsc{Trace}(z) \wedge \neg \textsc{Stage}(z)) \wedge \\ &\forall z (\textsc{Trace}(z) \rightarrow \neg \textsc{Stage}(z))\end{aligned}$$

(Nothing has more than one of the relevant properties. More explicitly, any two objects constructed from distinct types other than union are distinct; constructed objects are distinct from traces and stages; and traces are distinct from stages.)

(D25)

$$\begin{aligned}&\textsc{Settable}(x) \;\leftrightarrow \\ &\quad (\textsc{IsSet}(x) \;\vee\; \textsc{Individual}(x) \vee\; \textsc{IsPair}(x) \;\vee\; \textsc{Trace}(x))\end{aligned}$$

(A22) $\forall xx \forall x (\textsc{Set}(x : xx) \;\rightarrow\; \forall z (z \prec xx \;\rightarrow \textsc{Settable}(z)))$

(If some things construct a set, then each of them is settable.)

(A23)

$$\begin{aligned}&\forall xx \forall s (xx @@ s \;\wedge \forall x (x \prec xx \;\rightarrow\; \textsc{Settable}(x)) \;\rightarrow \\ &\qquad \exists x \textsc{Set}(x : xx))\end{aligned}$$

(For every plurality xx of settable objects existing at s, the set of xx exists.)

(A24) $\textsc{Injective}(c_{\text{set}})$

(The type set is injective. That is, a set is constructed from at most one plurality.)

(D26) $x \in y \ \leftrightarrow \ \exists yy(\textsc{Set}(y : yy) \wedge x \prec yy)$

(A25)

$$\exists xx \exists s \exists y(xx@@s \ \wedge \ \textsc{Given}(y) \ \wedge \ y \prec xx \ \wedge \ \forall x(x \prec xx \ \rightarrow \ \exists u \exists uu(\textsc{Set}(u : uu) \ \wedge \ \forall w(w \prec uu \ \leftrightarrow \ w = x) \ \wedge \ u \prec xx)))$$

(There are some xx such that xx exist at a stage, xx have a given as member and, whenever x is a member of xx, its singleton $\{x\}$ is also a member of xx.)

(D27) $\textsc{Summable}(x) \ \leftrightarrow \ \textsc{Individual}(x)$

(A26) $\forall xx \forall x(\textsc{Sum}(x : xx) \ \rightarrow \ \forall z(z \prec xx \ \rightarrow \textsc{Summable}(z)))$

(If some things construct a sum, then each of them is summable.)

(A27) $\forall z(z \prec xx \rightarrow \textsc{Summable}(z) \wedge z@s) \rightarrow \exists x \textsc{Sum}(x : xx)$

(For every plurality xx of summable objects existing at a stage s, the sum of xx exists.)

(A28) $\textsc{Sum}(x : xx) \wedge \forall u(u \prec xx \leftrightarrow u = y) \rightarrow x = y$

(The sum constructed from the singleton plurality of x is x.)

(A29)

$$\begin{aligned} &\textsc{Sum}(x : xx) \ \wedge \ \textsc{Sum}(y : yy) \wedge \\ &\quad \forall z_1(z_1 \prec xx \ \rightarrow \ (z_1 \prec yy \ \vee \exists zz(zz \preccurlyeq yy \ \wedge \ \textsc{Sum}(z_1 : zz)))) \wedge \\ &\quad \forall z_2(z_2 \prec yy \ \rightarrow \ (z_2 \prec xx \vee \\ &\qquad \exists zz(zz \preccurlyeq yy \ \wedge \ z_2 \prec zz \ \wedge \ \exists z_3(z_3 \prec xx \ \wedge \ \textsc{Sum}(z_3 : zz))))) \ \rightarrow \\ &\qquad\quad x = y \end{aligned}$$

(Suppose x is a sum of xx and y is a sum of yy. If the following two conditions are satisfied, then x is y:

1. every x among xx is either one of yy or is a sum of some zz among yy;
2. every y among yy is either one of xx or is one of some members of yy whose sum is among xx.

In other words, two pluralities, one obtained from the other by replacing some pluralities of objects with their respective sums, yield the same sum.)

(A30)

$$\forall x_1(x_1 \prec xx_1 \rightarrow \textsc{Given}(x_1)) \ \wedge \ \forall x_2(x_2 \prec xx_2 \rightarrow \textsc{Given}(x_2)) \wedge \textsc{Sum}(z_2 : xx_1) \ \wedge \ \textsc{Sum}(z_2 : xx_2) \rightarrow (z_1 = z_2 \ \rightarrow \ xx_1 \approx xx_2)$$

(If two sums of givens are identical, those givens must also be identical. That is, a sum can be constructed from givens xx_1 and also from givens xx_2, then xx_1 are the very same objects as xx_2.)

(D28) $x \leq y \ \leftrightarrow \ \exists yy(\textsc{Sum}(y : yy) \wedge x \prec yy)$

(D29) $\textsc{Atom}(x) \ \leftrightarrow \ \textsc{IsSum}(x) \ \wedge \ \forall y(y \leq x \ \rightarrow y = x)$

(A31)

$$\forall xx \forall x((\textsc{Left}(x : xx) \ \vee \ \textsc{Right}(x : xx)) \ \rightarrow \forall y \forall z(y \prec xx \ \wedge z \prec xx \ \rightarrow y = z))$$

(Any input plurality to the left or right constructor has no more than one member.)

(D30)

$$\textsc{Positionable}(x) \ \leftrightarrow (\textsc{IsSet}(x) \ \vee \ \textsc{Individual}(x) \ \vee \ \textsc{IsPair}(x) \ \vee \ \textsc{Trace}(x))$$

(A32)

$$\forall xx \forall x((\textsc{Left}(x : xx) \ \vee \ \textsc{Right}(x : xx)) \ \rightarrow \forall z(z \prec xx \ \rightarrow \textsc{Positionable}(z)))$$

(If some things construct a left or right object, then each of them is positionable.)

(A33)

$$\forall xx(\forall y \forall z(y \prec xx \ \wedge \ z \prec xx \ \rightarrow \ y = z) \wedge \forall z(z \prec xx \rightarrow \ \textsc{Positionable}(z)) \rightarrow \exists x_1 \textsc{Left}(x_1 : xx) \ \wedge \ \exists x_2 \textsc{Right}(x_2 : xx))$$

(If xx is singleton plurality whose member is positionable, then a left object and a right object with input xx exist.)

(A34) $\textsc{Injective}(c_{\text{left}}) \ \wedge \ \textsc{Injective}(c_{\text{right}})$

(Position types are injective. That is, a position object, left or right, is constructed from at most one plurality.)

(A35)

$$\forall xx \forall x(\textsc{Pair}(x : xx) \ \rightarrow \\ \exists y \exists z(y \prec xx \ \wedge \ \textsc{IsLeft}(y) \ \wedge z \prec xx \ \wedge \ \textsc{IsRight}(z) \wedge \\ \forall w(w \prec xx \ \rightarrow \ (w = y \ \vee \ y = z))))$$

(Any input plurality to the pair constructor has exactly two members: a left object and a right object.)

(A36)

$$\textsc{IsLeft}(x) \wedge \textsc{IsRight}(y) \rightarrow \\ \exists zz(\forall z(z \prec zz \leftrightarrow (z = x \vee z = y)) \wedge \\ \exists z \textsc{Pair}(z : zz))$$

(For every left object x and right object y, a pair with input x and y exists.)

(A37) $\textsc{Injective}(c_{\text{pair}})$

(The type pair is injective. That is, a pair is constructed from at most one plurality.)

(D31)

$$\textsc{Unionise}(xx, yy) \ \leftrightarrow \\ \forall y(y \prec yy \ \rightarrow \ \textsc{IsSet}(y)) \wedge \\ \forall x(x \prec xx \leftrightarrow \exists y(y \prec yy \ \wedge \ \exists zz(x \prec zz \ \wedge \ \textsc{Set}(y : zz))))$$

(A38) $\textsc{Union}(z : xx) \ \leftrightarrow \ \exists yy \ (\textsc{Set}(z : yy) \ \wedge \ \textsc{Unionise}(yy, xx))$

(z is the union of xx if and only if z is the set of some yy that "unionise" xx.)

(A39)

$$\forall w((\exists z \textsc{HasOutput}(w, z) \ \leftrightarrow \ \exists x \textsc{HasInput}(w, x)) \wedge \\ (\exists z \textsc{HasOutput}(w, z) \ \leftrightarrow \ \exists y \textsc{HasType}(w, y)))$$

(Any object has an output if and only if it has an input if and only if it has a type.)

(D32) $\textsc{Trace}(w) \ \leftrightarrow \ \exists z \textsc{HasOutput}(w, z)$

(A40)

$$\textsc{Trace}(w_1) \ \wedge \ \textsc{Trace}(w_2) \ \rightarrow \\ \forall x_1 \forall x_2 \forall y_1 \forall y_2 \forall z_1 \forall z_2((\textsc{HasOutput}(w_1, z_1) \ \wedge \ \textsc{HasOutput}(w_2, z_2) \wedge \\ \textsc{HasInput}(w_1, x_1) \ \wedge \ \textsc{HasInput}(w_2, x_2) \wedge \\ \textsc{HasType}(w_1, y_1) \ \wedge \ \textsc{HasType}(w_2, y_2)) \ \rightarrow \\ (z_1 = z_2 \ \wedge \ x_1 = x_2 \ \wedge \ y_1 = y_2 \ \leftrightarrow \ w_1 = w_2))$$

(Two traces are identical if and only if they have the same output, input, and type.)

(A41)

$$(\textsc{Construct}(z : xx, y) \ \wedge \ z@t \ \wedge \ \exists s(s \lhd t \ \wedge \ xx@@s)) \rightarrow$$
$$\forall x(x \prec xx \ \rightarrow \ \exists w\big(w@t \wedge$$
$$\textsc{HasOutput}(w, z) \ \wedge \ \textsc{HasInput}(w, x) \ \wedge \ \textsc{HasType}(w, y)))$$

(Suppose z is constructed from xx with type y, z exists at t, and xx exist at some stage before t. Then we also require at t the existence of appropriate traces. That is, for every member x of xx, there is at t a trace recording that the construction process has output z, input x, and type y.)

(A42)

$$\textsc{Trace}(w) \wedge w@t \rightarrow$$
$$\exists z \exists xx \exists x \exists y \exists s(\textsc{Construct}(z : xx, y) \wedge z@t \wedge xx@@s \wedge s \lhd t \wedge$$
$$x \prec xx \wedge \textsc{HasOutput}(w, z) \wedge \textsc{HasInput}(w, x) \ \wedge \ \textsc{HasType}(w, y))$$

(If a trace w exists at a stage t, then it must appropriately record some type of construction of an object existing at t from an input existing at some stage before t.)

(D33) $\textsc{Max}(s, t) \ \leftrightarrow \ \forall x(\textsc{ConstrFrom}(x, s) \rightarrow x@t)$

(A43) $\forall s \exists t \, \textsc{Max}(s, t)$

(Every stage has a maximal extension.)

G.2 Optional axioms

$$\textsc{Succ}(s, t) \rightarrow \textsc{Max}(s, t) \qquad (*)$$

H Future work

This project has achieved what it set out to do, that is, to establish that the formalisation of a unified constructional approach is feasible and consistent. As planned, we made several compromises to reach this goal as quickly as possible. We suggest that the next stage is to address these compromises as appropriate. Also, as often happens when one opens up a new area, further opportunities for improvement have been created. We have noted a few of these in the body of the report. Here we give a broad overview of the main ones we have identified so far. We distinguish between further short-term work that is essential to get the theory up to scratch and longer-term work that would merely improve or generalise.

H.1 Short term

Essential work needed in the short term includes the following tasks:

- set up an automated strategy to check, as far as possible, the consistency of the axioms inside CLIF;
- integrate the CCT with an existing formalisation of key aspects of the CCO in CLIF;
- investigate and implement a better solution for the representation of relations in the CCO and in the CCT. This should extend the notion of pair to tuples in general.

H.2 Long term

In the long term, we would aim to:

- assess the benefits of using a single constructor to mimic the two kinds of construction found in Lewis 1991;
- expand our constructional ontology to include deconstructors (e.g. one that decomposes a mereological objects into its proper parts);
- develop further our approach to constructing via CLAP profile as indicated in Appendix E.5;
- investigate more systematically the constructional possibilities in the CLAP classification;
- extend our approach to mereology, which is based on laws of identification and extremal clause, to other constructors, thus providing a general approach;
- examine different ways to effect "cross-identifications", i.e. identifications between outputs of different constructors;
- explore alternative formalisations of the constructional approach, such as dynamic formalisations and formalisations that eliminate stages in favour of their associated pluralities (for example, using critical plural logic as developed in Florio and Linnebo 2021, Chapter 12).

I Literature sources

I.1 Current situation with foundations

Set theory (e.g. Zermelo-Fraenkel set theory with the Axiom of Choice) and mereology (e.g. Classical Extensional Mereology, also known as General Extensional Mereology) are developed as separate axiomatic theories:

- Enderton 1977, *Elements of Set Theory*
- Simons 1987, *Parts: A Study in Ontology*
- Casati and Varzi 1999, *Parts and Places: The Structures of Spatial Representation*
- Varzi 2019, "Mereology"
- Cotnoir and Varzi 2021, *Mereology*

While these theories are usually developed within the same logical formalism (first-order logic), they are not part of a unified framework. In fact, their interaction raises a number of logical and philosophical issues:

- Lewis 1991, *Parts of Classes*
- Uzquiano 2006, "The Price of Universality"
- Florio and Linnebo 2021, *The Many and the One: A Philosophical Study of Plural Logic*

Ordered tuples are usually identified as set-theoretic constructions (see Enderton 1977, pp. 35–41), though they are occasionally developed as *sui generis*, e.g. in

- Bourbaki 1954, Section 2.1, *Théorie des ensambles. Éléments de Mathématique. Première partie, Livre I, Chapitres I, II.* (See also discussion in Kanamori 2003.)
- Tennant 2007, "Natural Logicism via the Logic of Orderly Pairing"

See also Pleitz 2017, Partridge, Cesare, et al. 2017, Partridge, Mitchell, Loneragan, et al. 2019, and Partridge, Mitchell, Loneragan, et al. manuscript referenced below.

I.2 History of constructional ontology

Many views in the history of philosophy have a constructional flavour or are outright examples of constructional approaches to ontology. Particularly relevant to our project are:

- Carnap 1928, *Der logische Aufbau der Welt* (first edition published in 1928)
- Goodman and Quine 1947, "Steps Toward a Constructive Nominalism"
- Goodman 1956, "A World of Individuals"
- Goodman 1958, "On Relations that Generate"

I.3 Current work on constructional ontology

New ideas by Kit Fine have given a fresh impetus to constructional ontology:

- Fine 1991, "The Study of Ontology"
- Fine 2002, *The Limits of Abstraction*
- Fine 2005, "Our Knowledge of Mathematical Objects"
- Fine 2010, "Towards a Theory of Part"

Fine's ideas have inspired recent work that is central to our project:

- De Cesare and Partridge 2016, "BORO as a Foundation to Enterprise Ontology"
- Partridge, Cesare, et al. 2017, "Developing an Ontological Sandbox: Investigating Multi-level Modelling's Possible Metaphysical Structures"
- Partridge, Mitchell, Loneragan, et al. 2019, "Coordinate Systems: Level Ascending Ontological Options"
- Partridge, Mitchell, Loneragan, et al. manuscript, "The Fantastic Combinations and Permutations of Co-ordinate Systems' Characterising Options: The Game of Constructional Ontology"
- Pleitz 2017, "Two Accounts of Pairs"

Our project provides a formal foundation for the constructional approach.

I.4 Other background work

The project draws on ideas in logic, philosophy of mathematics, and metaphysics developed in:

- Boolos 1971, "The Iterative Conception of Set"
- Boolos 1989, "Iteration Again"
- Florio and Nicolas 2015, "Plural Logic and Sensitivity to Order"
- Florio and Nicolas 2021, "Plurals and Mereology"
- Linnebo 2018, *Thin Objects: An Abstractionist Account*
- Linnebo 2013, "The Potential Hierarchy of Sets"
- Schaffer 2010, "Monism: The Priority of the Whole"
- Studd 2013, "The Iterative Conception of Set: A (Bi-)Modal Axiomatisation"

References

Bolton, Alexandra et al. (2018). *The Gemini Principles*. Tech. rep. Centre for Digital Built Britain.

Boolos, George (1971). "The Iterative Conception of Set". In: *Journal of Philosophy* 68.8, pp. 215–231.

— (1989). "Iteration Again". In: *Philosophical Topics* 17.2, pp. 5–21.

Bourbaki, N. (1954). *Théorie des ensambles. Éléments de Mathématique. Première partie, Livre I, Chapitres I, II.* Hermann.

Burgess, Gemma et al. (2020). *Flourishing Systems - Re-envisioning infrastructure as a platform for human flourishing*. Tech. rep. Centre for Digital Built Britain.

Carnap, Rudolf (1928). *Der logische Aufbau der Welt*. Weltkreis. Second edition: Meiner, 1961. Translated as *The Logical Structure of the World* by Rolf A. George. University of California Press, 1967.

Casati, Roberto and Achille Varzi (1999). *Parts and Places: The Structures of Spatial Representation*. MIT Press.

Chang, C. C. and H. Jerome Keisler (1990). *Model Theory*. North Holland.

Cotnoir, A. J. and Achille Varzi (2019). "Natural Axioms for Classical Mereology". In: *Review of Symbolic Logic* 12.1, pp. 201–209.

— (2021). *Mereology*. Oxford University Press.

De Cesare, Sergio and Chris Partridge (2016). "BORO as a Foundation to Enterprise Ontology". In: *Journal of Information Systems* 30.2, pp. 83–112.

Dickmann, M. A. (1975). *Large Infinitary Languages: Model Theory*. North-Holland.

Enderton, Hebert B. (1977). *Elements of Set Theory*. Academic Press.

— (2001). *A Mathematical Introduction to Logic*. Academic Press.

Fine, Kit (1991). "The Study of Ontology". In: *Noûs* 25.3, pp. 263–294.

— (1999). "Things and Their Parts". In: *Midwest Studies in Philosophy* 23.1, pp. 61–74.

— (2002). *The Limits of Abstraction*. Clarendon Press.

— (2005). "Our Knowledge of Mathematical Objects". In: *Oxford Studies in Epistemology* 1, pp. 89–110.

— (2010). "Towards a Theory of Part". In: *Journal of Philosophy* 107.11, pp. 559–589.

Florio, Salvatore and Øystein Linnebo (2018). "Logic and Plurals". In: *The Routledge Handbook of Collective Intentionality*. Ed. by Marija Jankovic and Kirk Ludwig. Routledge, pp. 451–463.

— (2021). *The Many and the One: A Philosophical Study of Plural Logic*. Oxford University Press.

Florio, Salvatore and David Nicolas (2015). "Plural Logic and Sensitivity to Order". In: *Australasian Journal of Philosophy* 93.3, pp. 444–464.

— (2021). "Plurals and Mereology". In: *Journal of Philosophical Logic* 50.3, pp. 415–445.

Gödel, Kurt (1964). "What is Cantor's Continuum Problem?" In: *Collected Works: Volume II: Publications 1938-1974*. Ed. by Solomon Feferman et al. Oxford University Press, 1990, pp. 176–188.

Goodman, Nelson (1956). "A World of Individuals". In: *The Problem of Universals: A Symposium*. Ed. by I. M. Bochenski, Alonzo Church, and Nelson Goodman. University of Notre Dame Press, pp. 13–31.

— (1958). "On Relations that Generate". In: *Philosophical Studies* 9.5/6, pp. 65–66.

Goodman, Nelson and W. V. Quine (1947). "Steps Toward a Constructive Nominalism". In: *Journal of Symbolic Logic* 12.4, pp. 105–122.

Hetherington, James and Matthew West (2020). *The pathway towards an Information Management Framework - A 'Commons' for Digital Built Britain*. Tech. rep. Centre for Digital Built Britain.

Hodges, Wilfrid (1977). *Logic: An Introduction to Elementary Logic*. Penguin Books.

Kanamori, Akihiro (2003). "The Empty Set, the Singleton, and the Ordered Pair". In: *Bulletin of Symbolic Logic* 9.3, pp. 273–298.

Kunen, Kenneth (1980). *Set Theory: An Introduction to Independence Proofs*. North-Holland.

Lewis, David (1991). *Parts of Classes*. Basil Blackwell.

Linnebo, Øystein (2013). "The Potential Hierarchy of Sets". In: *Review of Symbolic Logic* 6.2, pp. 205–228.

— (2018). *Thin Objects: An Abstractionist Account*. Oxford University Press.

NIC (2017). *Data for the Public Good*. Tech. rep. National Infrastructure Commission.

Partridge, Chris, Sergio de Cesare, et al. (2017). "Developing an Ontological Sandbox: Investigating Multi-level Modelling's Possible Metaphysical Structures". In: *Proceedings of MODELS 2017 Satellite Event co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017*. Ed. by Loli Burgueñoet al. Vol. 2019. CEUR Workshop Proceedings. CEUR-WS.org, pp. 226–234.

Partridge, Chris, Andrew Mitchell, Al Cook, et al. (2020). *A Survey of Top-Level Ontologies - To inform the ontological choices for a Foundation Data Model*. Tech. rep. Centre for Digital Built Britain.

Partridge, Chris, Andrew Mitchell, Michael Loneragan, et al. (2019). "Coordinate Systems: Level Ascending Ontological Options". In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, pp. 78–87.

— (manuscript). "The Fantastic Combinations and Permutations of Co-ordinate Systems' Characterising Options: The Game of Constructional Ontology".

Pleitz, Martin (2017). "Two Accounts of Pairs". In: *The Logica Yearbook 2016*. Ed. by Pavel Arazim and Tomáš Lávička. College Publications, pp. 201–221.

Schaffer, J. (2010). "Monism: The Priority of the Whole". In: *Philosophical Review* 119.1, pp. 31–76.

Schaffer, J. (2015). "What Not to Multiply Without Necessity". In: *Australasian Journal of Philosophy* 93.4, pp. 644–664.

Simons, Peter (1987). *Parts: A Study in Ontology*. Clarendon Press.

Studd, J. (2013). "The Iterative Conception of Set: A (Bi-)Modal Axiomatisation". In: *Journal of Philosophical Logic* 42.5, pp. 1–29.

Tennant, Neil (2007). "Natural Logicism via the Logic of Orderly Pairing". In: *Logicism, Intuitionism, Formalism: What has become of them?* Ed. by Sten Lindström et al. Springer Verlag, pp. 91–125.

Uzquiano, G. (2006). "The Price of Universality". In: *Philosophical Studies* 129, pp. 137–169.

Varzi, Achille (2019). "Mereology". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2019. `https://plato.stanford.edu/archives/spr2019/entries/mereology/`. Metaphysics Research Lab, Stanford University.

West, Matthew (2020). *The Approach to Develop the Foundation Data Model for the Information Management Framework*. Tech. rep. Centre for Digital Built Britain.

— (forthcoming). *Managing Shared Data*. Tech. rep. Centre for the Protection of National Infrastructure.

Acknowledgments

Lead Authors:

Salvatore Florio

Øystein Linnebo

Authors:

Chris Partridge

Martin Pleitz

Contributors:

Stefano Borgo

Al Cook

Kit Fine

Pierre Grenon

Anne Guinard

Graham Leach-Krouse

Liam McGee

Andrew Mitchell

Matthew West