Introduction to Constructional Ontology

Salvatore Florio, Øystein Linnebo

Department of Philosophy, Classics, History of Art and Ideas, University of Oslo, P.O. box 1020, Blindern N-0315 Oslo, Norway

Abstract

In constructional ontology, entities emerge by construction, that is, from the application of constructors to objects. We explore this approach to ontology, focusing on three modules: the constructors, the inputs to the constructors, and the constructional process. Our aim is to identify and assess some key theoretical choices arising in an ontology of this kind.

Keywords

foundational ontology, constructional ontology, set theory, mereology, plural logic, critical plural logic

1. Introduction

Kurt Gödel famously articulated a conception of set according to which 'a set is something obtainable from the integers (or some other well-defined objects) by iterated applications of the operation "set of" ' [1]. On this conception, the ontology of sets is generated through a constructional process. One begins with some objects—the *givens*—and the constructor "set of". The ontology is generated by successively applying the constructor to all available objects, thus producing larger and larger domains of sets. In each application, some objects are used as inputs to the constructor; the output is the set whose elements are precisely those objects. As the process unfolds, new constructional possibilities arise. For at each stage, new sets are constructed and thus become available as input for yet further instances of construction.

A similar approach can be deployed for other kinds of entities. For example, mereological sums too can be thought of as generated through a constructional process. In this case, one begins with some atomic entities and the constructor "sum of". Then all sums arise by means of applications of the constructor to available objects. In each application, some objects are used as inputs; the output is the mereological sum having at least those objects as parts. It is worth highlighting a disanalogy with the case of sets. A certain set can only be constructed from its zero or more elements. There are no other objects to which the "set of" constructor can be applied to obtain precisely that set. By contrast, one and the same sum can be obtained by applying "sum of" to different inputs. For example, a Lego figure can be the sum of its bottom and top halves, but also of its left and right halves.

One may be more ambitious and adopt a constructional approach to ontology *in general*, as advocated by Kit Fine [2, 3]. From this perspective, one's entire ontology is generated by means of a constructional process. Entities in the ontology are accepted *because* they can be constructed from accepted givens and constructors.

FOUST 2024

salvatore.florio@ifikk.uio.no (S. Florio); oystein.linnebo@ifikk.uio.no (Ø. Linnebo)

In this article, we explore this perspective on ontology, outlining its structure and the theoretical options it presents. We find it fruitful to regard a constructional ontology as organized around three "modules". They concern:

(i) the constructors;
(ii) the inputs to the constructors;
(iii) the constructional process.

We discuss each module at length below. The observed disanalogy between constructing sets and constructing sums already suggests some interesting and subtle differences that a general constructional ontology should be able to characterize.

Before taking a closer look at the modular structure of constructional ontology, let us briefly recapitulate some key motivations behind the approach—theoretical virtues that make the approach especially appealing (see [4] for a similar overview).

A major benefit of the constructional approach, emphasized by Gödel, is related to *consistency*. He observed [1, p. 180] that the constructional conception of set

> 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.

When managed appropriately, the constructional process can be a basis for consistency. Indeed, we can say something stronger. Not only is each type of construction *internally* consistent, different types of construction are also *mutually* consistent.

Another cluster of benefits has to do with *unification*. Theories that would normally be developed separately, such as set theory and mereology, arise in a unified way by means of a single constructional process. This has several advantages. Let us mention three. First, integrating separate theories might give rise to considerable difficulties. The unified development afforded by constructional ontology tackles integration problems at the outset and solves them, when possible, by design. Second, the constructional approach provides a uniform way of characterizing similarities and differences among types of objects on the basis of their constructional profile. For example, sets do, whereas sums do not, have a unique decomposition. We review further examples in later sections. Third, the approach promises a high degree of theoretical generality, providing a unified treatment of what may have appeared as disparate subject matters. As forcefully argued by Fine [3], one can, for instance, subsume both set theory and mereology under a general theory of part. In a constructional setting, to be part of an object is to be an input to the construction of that object. So set-theoretic membership and mereological parthood receive the same definition, exhibiting different ways in which an object can be part of another—being an element and being a mereological part.

Finally, we wish to note benefits related to *dependency* and *reduction*. Constructional ontology embodies a clear notion of dependence or ontological priority: an object x depends—at least in a weak sense—on another object y, if x can be constructed from y. A stricter form of dependence can be defined as irreversible weak dependence. Thus, a set depends (strictly) on its elements, and a sum depends (at least weakly) on its parts. The relations of dependency in the ontology are brought out by the constructional process. Ultimately, the ontology depends on the givens and is thus reducible to them.

2. Earlier work

Constructional ideas, it may be argued, have a very long history. For instance, suggestions to the effect that some entities can be constructed, assembled, or generated from others appear pervasive in the history of philosophy and mathematics (see [5] for a paradigmatic example). However, it is not so clear how close these suggestions are to the particular framework investigated in this article. Since our focus here is not historical, we will put aside interpretative questions and highlight only recent work that is directly connected to our discussion.

An important incarnation of constructional ontology is, without a doubt, the iterative conception of set. We already cited Gödel's famous remarks in [1]. Contemporary developments of the view, inspired by [6], play a significant role below (see, e.g., [7], [8], [9, Chapter 2], and [10]). Specifically, these contemporary developments inform various options available for the regimentation of the constructional process.

In present-day metaphysics, the constructional approach has been put back on the agenda by work of Fine mentioned above [2, 3]. But constructional ideas still need to be systematically explored. We hope that our discussion will encourage further contributions in this area.

We took some steps towards a more systematic development of a constructional framework in [4]. This technical report puts forward a new top-level ontology—the Core Constructional Ontology—for the Information Management Framework of the UK's National Digital Twin programme [11]. The report also extends and formalizes prior work on the constructional refactoring of the foundational ontology BORO [12, 13].[1]

3. Constructors

The first, and most obvious, module of a constructional ontology concerns the constructors. These are the "engines" of the ontology. They also determine the *types* of objects recognized. To each constructor there corresponds a type consisting of the objects generated by that constructor. We have already mentioned two examples of constructors: the set constructor ("set of") and the sum constructor ("sum of"). Other examples are readily available, such as constructors that generate cardinal numbers, ordinal numbers, lists of objects, and strings of characters.

In this section, we identify four important choices that arise when setting up a framework for constructors. They relate to different aspects of constructors and the way we may describe them. This set of choices is not exhaustive: we concentrate on them for reasons of space.

The first choice is whether or not to treat *constructors as entities*, that is, whether or not constructors are reified. One may allow quantification over constructors, which then become full citizens of the ontology—on a par with all other entities belonging to it. This option leaves open whether entities are objects or higher-order entities, hence whether the relevant form of quantification is first-order or higher-order. Alternatively, quantification over constructors may not be permitted. In that case, symbols for constructors operate like predicates and function symbols in first-order logic. By reifying constructors, we obtain greater expressive power. We can make generalizations about all forms of construction, for example, to the effect that some (such as "set of") are one-to-one, while others (such as "sum of") are not. Note further that

[1] For an introduction to BORO, see [14] and [15].

reifying constructors makes it possible to construct constructors as outputs of other constructors. Thus constructors themselves might emerge at some stage of the constructional process.[2]

The second choice is whether to treat constructors as *functional or relational*. In one case, a constructor is described by a functional expression in the language, such as a term-forming functional symbol. For instance, the set constructor may be represented by the functional symbol $f_{\text{set}}(\ldots)$, which can be used as a term in a predication such as $a \in f_{\text{set}}(\ldots)$. In the other case, a constructor is described by a relational predicate, governed by axioms laying out under what conditions some inputs are related to an output in the appropriate way. For sets, we use a predicate '$\textsc{Set}(xx, y)$' to express that y is a set constructed from the elements xx. (For notation and basic concepts of plural logic, see [16, Chapter 2].) We lay down that any objects xx, or at least any "suitable" objects xx, form a set y. Further, the natural criterion of identity asserts that, if xx_1 form y_1 and xx_2 form y_2, then $y_1 = y_2$ just in case xx_1 and xx_2 are the very same objects. A central difference between the relational approach and the functional one is that the former allows us to be more selective about when the relevant construction can be undertaken. As noted above, we may allow only objects that are in some sense suitable to form sets. Apart from this difference, however, we regard the choice between the two options as primarily one of convenience.[3]

The third choice is whether to provide an *explicit or recursive characterization* of constructors. In an explicit characterization, one provides necessary and sufficient conditions to identify the outputs of a constructor. Consider the case of sets. One lays out that, given some elements xx and some elements yy, the set of xx is identical to the set of yy if and only if the elements xx are the same as the elements yy. This pins down the identity conditions of constructed sets. By contrast, recursive characterizations offer sufficient conditions to identify the outputs of a constructor. Fine [3, pp. 573-576] discusses four conditions. Let us review them to gain a better understanding of recursive characterizations.

Let Σ be a constructor, and let us temporarily represent inputs as sequences of objects, following [3]. The principle of Collapse (C) states that, if the input to Σ is just x, then the output of the construction is x:

$$\Sigma(x) = x \tag{C}$$

The principle of Leveling (L) states the following. Start with two sequences, one obtained from the other by replacing some subsequences with the results of applying Σ to those subsequences. Then applying Σ to one sequence yields the same object as applying Σ to the other sequence.

$$\Sigma(\ldots, \Sigma(x, y, \ldots), \ldots, \Sigma(u, v,), \ldots) = \Sigma(\ldots, x, y, \ldots, u, v, \ldots) \tag{L}$$

Another principle is Absorption (A), which states that repetitions of an object in the input

[2] In fact, reifying constructors gives the option of introducing a "generic constructor", an operation that takes a specific constructor among its inputs, and that outputs objects of the corresponding type. For instance, to construct a set, one would feed the set constructor and the appropriate elements into the generic constructor (see [4, Sections 8.2 and 9.6] for an implementation of this setup). To avoid regress, the generic constructor itself would not be reified.

[3] This last point is defended in [17, Appendix 2B], where it is shown that each option can (in a precise technical sense) be imitated in the other—provided the functional approach is carried out in a free logic, which allows us to be selective about what is a permissible input to the constructor.

sequence of Σ are irrelevant to the result of the construction.

$$\Sigma(..., x, x, ..., y, y, ...) = \Sigma(..., x, ..., y, ...) \tag{A}$$

The last principle, Permutation (P), states that changing the order of the objects in the input sequence of Σ is irrelevant to the result of the construction.

$$\Sigma(..., x, y, z, ...) = \Sigma(..., y, z, x, ...) \tag{P}$$

By referring to these principles, one can provide a recursive characterization of different constructors (again, see [3]). One simply identifies which principles are satisfied by the constructor and thus constitute the constructor's "CLAP profile" (from the initials of the principles' names). For example, the CLAP profile of the set constructor is $\not{C}\not{L}$AP; that is, this constructor satisfies only Absorption and Permutation. The profile of the sum constructor is CLAP; that is, all principles are satisfied. This nicely captures some key differences between sets and sums. An interesting follow-up question is how to turn recursive characterizations into explicit ones.[4]

The fourth and final choice we consider here is whether or not constructors are given a *reductive characterization*. In a reductive characterization, the output of the constructor is specified in terms of material already available, such as objects already constructed or truths concerning prior stages of the constructional process. The explicit characterization of the set constructor given above is an example of a reductive characterization. Whether two applications of the set constructor result in the same output is determined entirely by relations between the inputs. In particular, it is determined by the identity of the two inputs. These objects are already available from prior stages of construction.

In a non-reductive characterization, by contrast, there is no guarantee that the identity or basic properties of the output can be specified in terms of available material. Suppose, for example, that for selected open formulas $\varphi(x)$, we can construct the property of being $\varphi(x)$, denoted $\lambda x.\varphi(x)$. We stipulate that this property applies to an object a just in case $\varphi(a)$. Suppose further that we construct the property s of self-application. We note that this property is not paradoxical. It is consistent both that it applies to itself and that it does not. Even so, we fail to reduce every claim about the application of s to some truth that was available prior to the construction. Does s self-apply? Applying the mentioned stipulation tells us that the answer is affirmative just in case s satisfies its own defining condition—which is precisely self-application! Thus, we fail to get a reduction.

Providing constructors with a reductive characterization has some important advantages. This makes it far easier to ensure that the construction is consistent. It also makes it possible to ensure that two legitimate forms of construction are mutually consistent (see [17, Chapter 9]).

4. Inputs

The second module of a constructional ontology concerns the inputs to the constructors—the "material" for the construction. In Section 3, when discussing the example of sets, we took the input to be some objects—or, as we will sometimes express it for convenience, a *plurality*. We

[4] See [4, Appendix E] for some initial work on this question.

start from *some objects*, the would-be elements, and apply the set constructor to them. The output is the set whose elements are precisely those objects.

A plurality—that is, some objects—is a very natural form of input for construction, and an especially compelling option in the case of sets. One can then simply rely on plural logic, a ready-to-use theory of one or more objects considered together, to regiment the behaviour of the inputs. There is *almost* a match made in heaven between plural logic and constructional ontology. Let us explain why.

Pluralities, as regimented in standard plural logic, are not sensitive to order or repetitions. The plurality of a, b, and c is the same the plurality of c, b, and a. It is also the same as the plurality of a, a, b, and c. So pluralities are very much like standard sets:

$$\{a, b, c\} = \{c, b, a\} = \{a, a, b, c\}$$

These features are ideal for some constructed entities, such as sets and mereological sums. Neither kind of entity is sensitive to order and repetitions. So the match is perfect.

However, the same features can be an obstacle for other constructed entities. Suppose we want to construct a list of objects, say the list of a, b, and c in that order: $[a, b, c]$. As an input, the plurality a, b, and c would not provide enough information for this construction, as it does not encode any order. So one needs to find a way of supplying the desired order.

One possibility is to use a *structured object* as an input. For example, one may consider feeding n-tuples to the list constructor. The list $[a, b, c]$ would result from the input $\langle a, b, c\rangle$, a triple with the appropriate order. But this option is obviously problematic in the present context. The structured objects we use as input should themselves be constructed. After all, the envisaged constructional approach to ontology is intended to be fully general.

A better option is to enrich the plural logic so as to represent some objects *in an order and perhaps with repetitions*. On this approach, we start with a theory of "structured pluralities", such as serial logic [18], which are not themselves reified.

This option suggests an even more general strategy. One could supplement an input plurality with "extra information", not only encoding order and repetitions, but also supplying *conceptual material*. Different notions of *embodiment*, as developed by Fine ([19], see also [20]), could be used to develop this idea. Let us consider an example. A *rigid embodiment* is an object combining some things xx with a "form", a relation R among xx—for instance some flowers in the characteristic spatial relation of a bouquet yield a bouquet of flowers. Similarly, a constructor could take a plurality of things together with a form. A list might then be constructed by inputting a plurality together with a relation that orders the members of the plurality in the desired way.

A more conservative but less general option is to stick with ordinary plural logic, where pluralities are set-like (apart from not being reified), and to let order emerge from construction. To this end, one introduces appropriate constructional devices. Let us illustrate the idea by working through an example [4, Section 9.11]. Suppose we wish to construct ordered pairs using a *pair constructor*. We start with a and b. Our target is the pair $\langle a, b\rangle$, with a as left coordinate and b as right coordinate. The required order could be obtained by means of auxiliary constructors, call them the *left constructor* and the *right constructor*. The left constructor takes as input a singleton plurality, in this case the plurality of a only, and yields a "left object", namely a

as left coordinate. The right constructor behaves similarly, yielding a "right object", namely b as a right coordinate. Once the appropriate left object and right object have been constructed, their doubleton plurality serves as input to the construction of the ordered pair. While the plurality is unordered, the order characterizing the pair is encoded in the constructional history of the pair. It can be retrieved accordingly. To determine the left coordinate, one simply identifies the input of the left object used to generate the ordered pair.

Generalizing, order and other features of constructed objects could be obtained by means of auxiliary constructors tied to roles. In our example, the roles are being the first coordinate and being the second one.

It is worth mentioning that supplying the order through enriched pluralities or as extra information can, at least in some cases, be avoided. One can instead rely on the syntax of the language, since order can also be encoded through the argument structure of a function symbol for the constructor. For instance, the pair constructor could be represented by the symbol f_{pair} taking two individual terms as separate arguments. The order of the pair would then be reflected in the order of the arguments following the function symbol. So the ordered pair $\langle a, b\rangle$ would correspond to the expression '$f_{\text{pair}}(a, b)$', whereas the ordered pair $\langle b, a\rangle$ would correspond to the expression '$f_{\text{pair}}(b, a)$' (for a development of this approach, see [21]).

This syntactic strategy can avoid enriched pluralities, order as explicit extra information, and even pluralities altogether. However, it has limited scope. In standard formal languages, terms and formulas have finite length. Thus function symbols take only a finite number of arguments. This means that constructions relying on an infinite number of ordered inputs, say to build an infinite list, cannot be represented in this way. Even when order is not required, as in the case of sets, the syntactic strategy is not sufficient. Constructing an infinite set would still require an infinite series of arguments. So the strategy must, in general, be combined with forms of construction permitting inputs that represent infinite collections. All in all, some form of plural logic seems key to securing the availability of such collections.

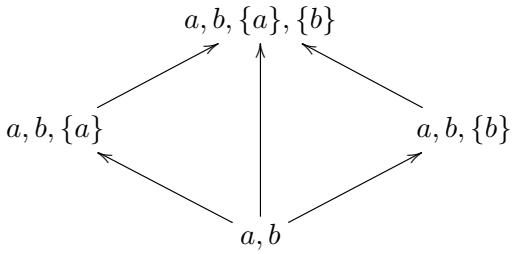
5. Process

The third and last module concerns the structure of the constructional process. While extant work on constructional ontology has begun to examine some of the main options for constructors and inputs, this third module is usually left unexplored.

A first set of questions arises in connection with the *structure of the constructional process*. A natural assumption is that the process has a linear structure and, in fact, a well-ordered one. One starts with the givens. One then move from one stage to the next by performing some construction. There might be infinite stages, that is, stages that have preceding stages but no immediately preceding one. As a model of this default setting, one can think of the structure of stages in the iterative conception of set [6].

However, linearity is not forced. It is perfectly consistent to assume that the constructional process has a branching structure. This would reflect the possibility of developing the constructional process in different ways. Suppose, for example, that at a given stage we have two objects a and b. We might want to countenance three alternative constructional possibilities. In one possibility, we construct the singleton of a but not that of b. In another, we construct the

singleton of b but not that of a. The third possibility is to construct both singletons at the same time. A branching structure allows us to represent these possibilities, each of them immediately accessible from the given stage.



If branching is permitted, one faces the further question whether branches must always converge. Convergence ensures that alternative constructional possibilities, such as those in the simple example just discussed, are merely choices about the order in which we reach the same constructional outcome. Returning to that example, we have a choice whether to construct (i) $\{a\}$ before $\{b\}$, (ii) $\{b\}$ before $\{a\}$, or (iii) $\{a\}$ and $\{b\}$ simultaneously. In any case, both singletons will eventually be constructed. Thus the constructional possibilities converge.

Without convergence, there might be genuine constructional alternatives: constructional choices that forever forestall others. This kind of situation is common in concrete domains, where incompossible objects abound. We have an ingredient that is required by two different recipes. We decide to use it for one recipe, forsaking the possibility of using it for the other. We bring about one dish at the expense of the other. It might well be that one's constructional ontology encompasses analogous alternatives. Even in the abstract domain, there are examples of divergent constructions. Suppose we are given a free choice of how to extend a finite sequence of objects (see, e.g., [22]). Then, choosing to extend with a is incompatible with choosing to extend with a distinct object b. In cases of this form, one adopts a matching order-theoretic structure for the constructional process.

Usually, one thinks of a constructional ontology as starting from some givens or the empty domain. The idea is that there is a single point from which the constructional process unfolds. However, one could also embrace a different picture, one according to which there are many alternative starting points. Each starting point gives rise to a constructional history. The result is a multiplicity of histories that could overlap and even converge.

Our example of branching assumed that some constructions might be delayed. Even though it is possible to construct both $\{a\}$ and $\{b\}$, it is also possible to construct $\{a\}$ before $\{b\}$ or vice versa. This means that there is a stage where only one of the two singletons exists. The assumption may be rejected. Instead, one may assume *maximality*: all constructional possibilities are realized as soon as possible. Maximality sanctions that $\{a\}$ and $\{b\}$ exist at a stage immediately following the first stage at which a and b exist. There is never any needless delay—for the construction of nested sets is delayed, but not needlessly so.

Note that maximality does not rule out branching. If some objects are incompossible from the constructional point of view, then one will need to choose among them even when the construction is as quick as it can be. In this case, we have maximality as well as branching. This also shows that maximality does not imply convergence. If the objects are incompossible, some

alternative possibilities can never be brought together. The branches do not converge.

The questions just discussed arise in connection with the structure of the constructional process. Another important set of questions has to do with *alternative ways of representing the constructional process*.

So far, we have often referred to *stages* of the constructional process. We relied on them to illustrate a number of key ideas, from that of a reductive characterizations of constructors to the notion of maximality. While stages can undoubtedly play an important heuristic role, it is wide open how they should be represented.

A straightforward option is to reify stages, treating them as primitive objects in the ontology. An example of this setup can be found in the classical development of the iterative conception of set [6], where stages are *sui generis* objects populating the domain of the theory alongside sets. For instance, one might postulate that stages are well ordered and then describe what exists at each stage. In effect, one develops the constructional ontology as a stage theory in the sense of the iterative conception of set.

On this picture, however, stages are neither givens nor constructed objects. They are auxiliary entities forming, one might say, the infrastructure of the constructional process. So it might be tempting to look for ways of avoiding them altogether. We discuss three possibilities. One uses modal logic. Two others use plural logic.[5]

Stage-theoretic structures of the kind relevant here can be described by means of modalities. We may think of each stage of the constructional process as a possible world. We can then use the modal operators '$\Diamond$' and '$\Box$' to theorize about constructional possibilities and what will hold no matter what we construct, respectively. For example, to express that, no matter what objects xx we will ever have constructed, these can be used to construct a set, we use:

$$\Box\forall xx\, \Diamond\exists y\, \textsc{Set}(xx, y)$$

Further, by making appropriate choices about the modal logic governing these operators, we can express key assumptions about the global structure of the constructional process. (Readers who are unfamiliar with modal logic may skip the rest of this paragraph.) A natural starting point is the modal logic S4, representing the fact that extension by construction is reflexive and transitive. Less obviously, we can express that the constructional process is convergent (in the sense explained above) by adopting the following modal axiom:

$$\Diamond\Box\varphi \rightarrow \Box\Diamond\varphi \tag{G}$$

The use of modal operators is not obligatory, however. Let us think about the minimum of expressive resource we require. Stages have domains. The domain of a stage encompasses the objects that exist at that stage. Further, assuming that the construction is deterministic—that is, that the atomic properties of the constructed objects are determined by properties of the constructors and their input—some objects, once constructed, will never differ as to their atomic properties. This suggests another way to avoid primitive stages. One could let a stage be represented directly by the plurality of objects that exist at that stage. Thus pluralities can

[5]Yet another option is Fine's "procedural postulationism" [23], which uses an imperatival logic to express and reason about postulations.

play the role of stages, which no longer need to be reified. Talk of pluralities is regimented, as discussed above, by means of plural logic.

Consider, for example, the initial stage. Only the givens exist at that stage. So we could let the initial stage simply be the plurality of givens. There is no need to postulate a *sui generis* object, a stage, over and above the givens themselves. The next "stage" of the constructional process can also be represented by another plurality, the plurality of objects constructible from the givens. And so on for all stages. Let us call these special pluralities "stage pluralities".

Traditional plural logic includes an extremely permissive principle of existence for pluralities: any meaningful condition defines a plurality, provided the condition is satisfied by at least one object. That is, for any condition φ, if there is a φ, then the plurality of φs exists. For many forms of construction, this entails the existence of a plurality that is not contained in a stage plurality. To see why, consider again the case of sets. Suppose that the construction has the structure of the stages in the iterative conception of set. Larger and larger domains are built by constructing, at each stage, all possible sets based on objects available at that stage. Traditional plural logic licenses the existence of the *plurality of all sets*. However, on pain of contradiction, this plurality cannot be contained in a stage plurality. If it were, we would be able to use the members of the plurality as inputs for the construction of a new set. This would be the set of all sets, whose existence is refutable in this setting.

We are confronted with a choice. One option is to retain traditional plurality logic and accept the existence of pluralities (such as the plurality of all sets) that are not contained in a stage plurality. This, in turn, means that not every plurality is eligible to serve as input to the constructors. A plurality can serve as input to a constructor only if it is contained in some stage plurality.

An alternative option is to use a more restricted plural logic that licences only pluralities contained in stage pluralities and thus ensures that *every* plurality can serve as input to the constructors. This can be achieved by using *critical plural logic* [24, 16], which appears a better fit for constructional ontology. Critical plural logic is more cautious in what pluralities it asserts to exist, postulating only pluralities that can either serve as stage pluralities or be contained in stage pluralities. Thus, all the work previously done by stages can now be done by pluralities; there is not even a need for a new predicate true of all and only stage pluralities. So the logic can serve as a "calculus of stages". Let us illustrate this claim.

Critical plural logic permits pairwise unions of pluralities. Whenever there are xx and yy, there are zz whose members are all and only the members of xx and yy. In constructional terms, this amounts to assuming that any two stages converge. A broader principle of generalized union, also sanctioned by critical plural logic, provides a stronger form of convergence. The principle states the following. Suppose there are some pluralities, each associated with a unique object—a "tag" for the plurality. Then, if the tags belong to a common plurality, the pluralities can be "unionized". That is, there is another plurality whose members are all and only the members of the given pluralities. From the constructional point of view, this amounts to the convergence of a range of stages, subject to the condition that their tags co-exist at some stage.

6. Advantages of our modular approach

We have explored a range of options available in constructional ontology. The options belong to three main modules around which the ontology can be organized: constructors, inputs, and constructional process. In this section, we would like to highlight some of the advantages of our modular structure. This structure enables us to choose modules to fit our needs and interests. It becomes easy to assemble an "object factory" that is tailored to our needs. Let us provide some examples. Each example shows how different ontologies can arise by varying one module, while keeping the two remaining modules fixed.

First, we may replace one "engine" in our "factory" with another. Less metaphorically, we may swap one constructor for another while retaining everything else (that is, the input to the construction and the global structure of the constructional process). Instead of constructing sets, for instance, we may construct mereological sums. Suppose two parties agree about the inputs and about the constructional process. Constructors take as inputs any plurality of objects that are available at one and the same stage. The constructional process unfolds, as before, along an infinite, well-ordered sequence of stage. The two parties disagree only on which constructors they admit: A accepts only the set constructor, while B accepts only the sum constructor. So A and B end up with two different foundational theories: Zermelo-Fraenkel set theory and atomistic classical extensional mereology [4, Section 10].

Second, we may keep the constructor(s) and the global structure fixed but swap the kind of input we provide to the constructor. For example, consider the constructor that turns any plurality of objects, possibly with some structure, into an object. We may call this constructor *plain reification*, since two outputs will be identical just in case the inputs consist of the same objects with the same structure (if any). Let us now vary the kind of input we feed into this constructor. We can feed it either ordinary pluralities, which are insensitive to order and repetition, or structured pluralities that *are* sensitive to order and/or repetition. As we vary the input, we obtain different forms of output, such as sets, multisets (which are sensitive to repetition but not order), and sequences (which are sensitive to both order and repetition).

Third, we can hold our choice of constructors and input fixed but vary the global structure. This presents us with various options. For one thing, we can choose a way to represent and theorize about the constructional process: a stage theory based on *sui generis* stages, a modal approach, or an approach based on traditional or critical plural logic.

For another, we can choose which assumptions to make about the global structure of the constructional process. Let us mention three choices. One choice is to allow the process be to run infinitely far or require that every stage be reachable in finitely many steps. Suppose we are constructing sets. Then, on the former option, we end up with a rich Cantorian universe satisfying the axioms of Zermelo-Fraenkel set theory [4, Section 10.2]. On the latter option, by contrast, we end up just with hereditarily finite sets—in essence, finite sets, whose elements are also finite and such that their elements, in turn, are yet again finite, and so on "all the way down".

A second choice is whether the constructional process is linear or branching. The linear option has the advantage of being pleasingly simple. The branching option, however, yields more fine-grained information. For instance, it provides information about dependencies among the objects we are constructing. Suppose it is impossible to construct b without first constructing

a. Then b (strictly) depends on a.

If we permit branching, a third choice concerns the assumptions one makes about the branches. This choice becomes stark when we start with a finite number of givens. Then we can allow:

(a) only pairwise convergence, that is, any two branches have a common extension;
(b) countable convergence, as can be done in critical plural logic, using its generalized union principle explained above;
(c) unrestricted convergence, as is required to bring together all the possible sets of givens and thus, in turn, constrain the uncountably infinite power set of the set of givens.

Here, too, we believe that having a choice is good: the options we have presented correspond to views in the foundations of mathematics. The first option allows only the construction of hereditarily finite sets. The second position corresponds to "countabilism", which allows the construction of all and only hereditarily countable sets. The final option is the orthodox Cantorian one, which permits the construction of uncountable sets.[6]

7. Conclusion

The constructional approach to ontology has a number of appealing features: it can be a basis for consistency, it has a high degree of unification, and it embodies a clear notion of ontological priority. Implementing the approach presents us with various theoretical choices. As a result of these choices, a wide range of different ontologies can be developed within the constructional framework, reaping the mentioned benefits.

However, a systematic investigation of the constructional approach is still lacking, and uses of the approach in applied ontology are so far limited.[7] Thus it seems fair to say that the full theoretical and practical potential of the approach is far from having been realized.

Let us conclude by highlighting some topics and open research questions we deem particularly interesting. In Section 3, we mentioned that constructors can be given an explicit characterization, but they can also be given a recursive characterization. The relation between the two characterizations is worth investigating. One question broached above is how to turn recursive characterizations into explicit ones.

The theoretical choices described in this article make the constructional approach very flexible. Indeed, as observed, the approach can be deployed to obtain many different ontologies. One might explore whether the approach can be made even more flexible. For instance, *deconstructors* might be countenanced in addition to constructors. To give an example, a set deconstructor takes a set as input and outputs the elements of the set. Then one may ask whether this addition increases the strength of the ontology and, if so, how.

Theorizing about propositions, properties, and relations (PPRs) is notoriously prone to paradox. We noted that, if appropriately managed, construction can be a basis for consistency. So it is natural to investigate whether the constructional approach can be used to formulate an

[6] Finitism and Cantorian realism are familiar options. See [25] for a defense of the less familiar, intermediate option of countabilism.

[7] One exception is the work done in connection with BORO, referenced in Section 2.

adequate theory of PPRs.[8] This is a challenging task. However, given the importance of PPRs in philosophy, linguistics, psychology, and beyond, it might also be an extremely rewarding one. One stumbling block was illustrated in Section 3 with the example of self-application. This showed that it is hard to ensure reducibility, that is, to guarantee that the identity or basic features of a property can be specified in terms of available material. But it is precisely reducibility that can be a great aid to consistency.

We surveyed some alternative ways of representing the constructional process (Section 5). It seems worthwhile to study the implementation of constructional ideas in an even wider range of frameworks. Obvious candidates include systems of a broadly constructional flavour, such as various forms of constructive type theory and dynamic logics.

Acknowledgments

[Acknowledgments]

References

[1] K. Gödel, What is Cantor's continuum problem?, 1964. *Collected Works: Volume II: Publications 1938-1974*. Ed. by Solomon Feferman et al. Oxford University Press, 1990, pp. 176–188.
[2] K. Fine, The study of ontology, Noûs 25 (1991) 263–294.
[3] K. Fine, Towards a theory of part, Journal of Philosophy 107 (2010) 559–589.
[4] S. Florio, Ø. Linnebo, Core Constructional Ontology: The Foundation for the Top-Level Ontology of the Information Management Framework, Technical Report, 2022.
[5] R. Carnap, Der logische Aufbau der Welt, 1928. Weltkreis. Second edition: Meiner, 1961. Translated as *The Logical Structure of the World* by Rolf A. George. University of California Press, 1967.
[6] G. Boolos, The iterative conception of set, Journal of Philosophy 68 (1971) 215–231.
[7] J. Studd, The iterative conception of set: A (bi-)modal axiomatisation, Journal of Philosophical Logic 42 (2013) 1–29.
[8] Ø. Linnebo, The potential hierarchy of sets, Review of Symbolic Logic 6 (2013) 205–228.
[9] L. Incurvati, Conceptions of Set and the Foundations of Mathematics, Cambridge University Press, 2020.
[10] T. Button, Level theory, part 1: Axiomatizing the bare idea of a cumulative hierarchy of sets, Bulletin of Symbolic Logic 27 (2021) 436–460.
[11] J. Hetherington, M. West, The pathway towards an Information Management Framework - A 'Commons' for Digital Built Britain, Technical Report, Centre for Digital Built Britain, 2020.
[12] C. Partridge, S. de Cesare, A. Mitchell, F. Gailly, M. Khan, Developing an ontological sandbox: Investigating multi-level modelling's possible metaphysical structures, in: L. B. noet al (Ed.), Proceedings of MODELS 2017 Satellite Event co-located with ACM/IEEE 20th

[8] See [26] and [27] for some relevant ideas.

International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017, volume 2019 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 226–234.
[13] C. Partridge, A. Mitchell, M. Loneragan, H. Atkinson, S. de Cesare, M. Khan, Coordinate systems: Level ascending ontological options, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2019, pp. 78–87.
[14] C. Partridge, Business Objects: Re-Engineering for Re-USe, Butterworth-Heinemann, 1996.
[15] S. De Cesare, C. Partridge, BORO as a foundation to enterprise ontology, Journal of Information Systems 30 (2016) 83–112.
[16] S. Florio, Ø. Linnebo, The Many and the One: A Philosophical Study of Plural Logic, Oxford University Press, 2021.
[17] Ø. Linnebo, Thin Objects: An Abstractionist Account, Oxford University Press, 2018.
[18] S. Hewitt, The logic of finite order, Notre Dame Journal of Formal Logic 53 (2012) 297–318.
[19] K. Fine, Things and their parts, Midwest Studies in Philosophy 23 (1999) 61–74.
[20] G. Uzquiano, Groups: Toward a theory of plural embodiment, Journal of Philosophy 115 (2018) 423–452.
[21] M. Pleitz, Two accounts of pairs, in: P. Arazim, T. Lávička (Eds.), The Logica Yearbook 2016, College Publications, 2017, pp. 201–221.
[22] E. Brauer, The modal logic of potential infinity: Branching versus convergent possibilities, Erkenntnis 87 (2020) 2161–2179.
[23] K. Fine, Our knowledge of mathematical objects, Oxford Studies in Epistemology 1 (2005) 89–110.
[24] S. Florio, Ø. Linnebo, Critical plural logic, Philosophia Mathematica 28 (2020) 172–203.
[25] D. Builes, J. M. Wilson, In defense of Countabilism, Philosophical Studies 179 (2022) 2199–2236.
[26] G. Sbardolini, Aboutness paradox, Journal of Philosophy 118 (2021) 549–571.
[27] Ø. Linnebo, S. Shapiro, Predicativism as a form of potentialism, Review of Symbolic Logic 16 (2023) 1–32.