Automating Leibniz's Theory of Concepts[*]

Jesse Alama
Vienna University of Technology
alama@logic.at
and
Paul E. Oppenheimer
Stanford University
paul.oppenheimer@stanford.edu
and
Edward N. Zalta
Stanford University
zalta@stanford.edu

Abstract

Our computational metaphysics group describes its use of automated reasoning tools to study Leibniz's theory of concepts. We start with a reconstruction of Leibniz's theory within the theory of abstract objects (henceforth 'object theory'). Leibniz's theory of concepts, under this reconstruction, has a non-modal algebra of concepts, a concept-containment theory of truth, and a modal metaphysics of complete individual concepts. We show how the object-theoretic reconstruction of these components of Leibniz's theory can be represented for investigation by means of automated theorem provers and finite model builders. The fundamental theorem of Leibniz's theory is derived using these tools.

1 Introduction

The computational metaphysics group at Stanford University's Metaphysics Research Lab has been engaged in a project of implementing object theory, i.e., the axiomatic theory of abstract objects [19, 20], in a first-order (with identity) automated reasoning environment. The first efforts [4] established that Prover9 could be used to represent and derive the theorems that govern possible worlds [22] and Platonic Forms [17]. Our focus over the past several years has been to represent and derive the theorems in [24], a paper that shows how to apply object theory to derive Leibniz's non-modal 'calculus' of concepts, his containment theory of truth, and his modal metaphysics of 'complete individual concepts' (defined below). Leibniz's theory of concepts is still interesting today, for several reasons. The calculus of concepts was one of the first axiomatizations of semi-lattices; his containment theory of truth anticipated work on generalized quantifiers [12]; and the modal metaphysics of complete individual concepts shows how to reconcile Lewis's 'counterpart' interpretation of quantified modal logic [11] with the standard (Kripke) interpretation [8]. Though these features were explored in detail in [24], we rehearse the basics below, when we sketch the background of our work with automated reasoning tools.

In this paper, we describe not only our results of using E, Vampire, and Paradox to automate the theory of concepts, but also the obstacles, insights, and other interesting issues that arose during the course of our investigations. The use of automated deduction tools allowed us to find out interesting things about representing richer logics in $\mathsf{FOL}_=$ (first-order logic with identity), and about the reasoning needed to derive Leibniz's results. In what follows, we focus on the core principles of Leibniz's theory of concepts, and we make no attempt to derive or explain the many other ambitious theories Leibniz attempted to develop (such as his theodicy for explaining the presence of evil, or his plan for world peace by dissolving the ideological case for religious wars).

2 Overview of Object Theory and Two Applications

2.1 The Basics of Object Theory

Object theory [19, 20, 22, 24] is an axiom system formalized in a syntactically second-order modal predicate calculus in which there is a primitive 1-place predicate $E!$ ('concreteness'). (Identity is not primitive; see below.) The system uses complex terms of two kinds, namely, definite

[*] Published in A. Felty and A. Middeldorp (eds.), *Automated Deduction – CADE 25: Proceedings of the 25th International Conference on Automated Deduction* (Lecture Notes in Artificial Intelligence: Volume 9195), Berlin: Springer, pp. 73–97.

descriptions and λ-expressions, where the latter denote relations rather than functions. The distinguishing feature of object theory is that the language uses two kinds of atomic formulas:

- $F^n x_1 \ldots x_n$ (for $n \geq 0$)
  - These are the atomic ('exemplification') formulas of standard FOL. When $n \geq 1$, these are read as "$x_1, \ldots, x_n$ *exemplify* F^n" and when $n = 0$, as "F^0 is true".
- xF^1
  - These are new, monadic atomic ('encoding') formulas. These are read as "x *encodes* F^1" and we henceforth drop the superscript on the F^1.

In what follows, we'll often substitute the variables $x, y, z, \ldots$ for $x_1, x_2, \ldots$ and say that they range over *objects* or *individuals*, while the variables $F^n, G^n, \ldots$ range over n-place *relations* (recall that the language of object theory is second-order).

Encoding formulas are best explained by first introducing two defined predicates used in object theory: *being ordinary* ('$O!$') and *being abstract* ('$A!$'). An ordinary object is one that is possibly concrete (i.e., $O!x \ =_{df} \ \Diamond E!x$), whereas an abstract object is one that couldn't be concrete (i.e., $A!x \ =_{df} \ \neg\Diamond E!x$). Note that these definitions partition the domain of objects.

Intuitively, ordinary objects are the kinds of things we might encounter in experience. They only *exemplify* their properties, and the standard formulas of the classical predicate calculus are sufficient to represent claims about which properties and relations ordinary objects exemplify or stand in.

But abstract objects aren't given in experience; nor is there a Platonic heaven out there containing abstract objects waiting to be discovered. Instead, abstract objects are identified by the properties by which we conceive of them. For example, mathematical objects are abstract objects; the only way we can get information about them is by way of our theories of them. We use encoding formulas to indicate the properties F by which we theoretically conceive of an abstract object x. For example, where κ is a uniquely defined object term in some mathematical theory T, object theory identifies κ as the abstract object that encodes all and only the (mathematical) properties attributed to κ in T (either by assumption or by proof). Though mathematical objects are identified by their *encoded* properties, they also *exemplify* non-mathematical properties. So whereas the number 0 of Peano Number Theory encodes such mathematical properties as $[\lambda x\, x < 1]$, $[\lambda x\, \forall y(y + x = y)]$, etc., it exemplifies non-mathematical properties, such as *being thought about by mathematician z*, *being abstract*, *not being a building*, etc. The first of the latter group is contingently exemplified by 0 (depending on the mathematician), while the second and third are necessarily exemplified by 0. However, the properties encoded by 0 *constitute* 0, since they are even more important to the identity of 0 than its necessarily exemplified properties.

We mentioned that there are two kinds of complex terms, definite descriptions and λ-expressions. Descriptions denote individuals, while the n-place λ-expressions denote n-place relations. Definite descriptions have the form $\imath x\phi$, and are to be read: the x in fact such that ϕ. In the modal contexts of object theory, these terms are interpreted rigidly (i.e., semantically, $\imath x\phi$ denotes the unique object that satisfies ϕ, if there is one, at the actual world of the model). λ-expressions have the form $[\lambda x_1 \ldots x_n\, \phi]$. The principles of α-, β-, and η-conversion for λ-expressions are assumed as axioms, though β-conversion is taken to be an equivalence and not an equation. It is important to note that λ-expressions obey the restriction that ϕ have no encoding subformulas. This is to avoid a Russell-style paradox.[1] As previously mentioned, λ-expressions are to be understood relationally in object theory, not functionally. That is, $[\lambda x_1 \ldots x_n\, \phi]$ doesn't denote an n-ary function, but rather an n-place relation, i.e., an element of a primitive domain of n-place relations.[2]

[1] If we were to allow a predicate of the form $[\lambda x\, \exists F(xF \;\&\; \neg Fx)]$, then an abstract object that encodes such a property would exemplify the property if and only if it doesn't. The paradox is avoided by banishing encoding from λ-expressions.

[2] Note that since λ-expressions may not contain encoding subformulas, the comprehension principle for relations derivable from β-Conversion becomes similarly restricted. β-Conversion asserts that $[\lambda x_1 \ldots x_n\, \phi]y_1 \ldots y_n \equiv \phi^{y_1,\ldots,y_n}_{x_1,\ldots,x_n}$. We can universally generalize on each of the y_is to obtain:

$$\forall y_1 \ldots \forall y_n([\lambda x_1 \ldots x_n\, \phi]y_1 \ldots y_n \equiv \phi^{y_1,\ldots,y_n}_{x_1,\ldots,x_n})$$

Then we apply the Rule of Necessitation and existential generalization to obtain:

> $\exists F^n \Box \forall y_1 \ldots \forall y_n(F^n y_1 \ldots y_n \equiv \phi)$, provided F^n doesn't occur free in ϕ and ϕ has no encoding subformulas.

This comprehension principle doesn't guarantee that there are any relations definable in terms of encoding predications.

The two main principles governing encoding predications and abstract objects are a comprehension principle and an identity principle for objects. The comprehension principle asserts: for any formula ϕ that places a condition on properties, there is an abstract object that encodes all and only the properties F satisfying (in Tarski's sense) ϕ. The comprehension principle is formalized as a schema:

> $\exists x(A!x \,\&\, \forall F(xF \equiv \phi))$, where ϕ is any formula in which x doesn't occur free

Here are some instances:

$$\exists x(A!x \,\&\, \forall F(xF \equiv Fa))$$

$$\exists x(A!x \,\&\, \forall F(xF \equiv F = R \vee F = S))$$

$$\exists x(A!x \,\&\, \forall F(xF \equiv \text{In Peano Number Theory}, F0))$$

These respectively assert the existence of an abstract object that: (a) encodes just the properties exemplified by object a; (b) encodes just the properties R and S, and (c) encodes just the properties attributed to 0 in Peano Number Theory.[3]

Identity is not primitive in object theory. Rather, it is defined for both objects and relations. Since the definition of relation identity doesn't play a role in what follows, we discuss only the definition of identity for objects. We define: $x = y$ iff either x and y are ordinary objects that necessarily exemplify the same properties or x and y are abstract objects that necessarily encode the same properties, i.e.,

$$x = y \ =_{df}$$
$$(O!x \,\&\, O!y \,\&\, \Box\forall F(Fx \equiv Fy)) \vee (A!x \,\&\, A!y \,\&\, \Box\forall F(xF \equiv yF))$$

Thus, if we know that objects x and y are abstract, we have to show that they necessarily encode the same properties to show that they are identical. With identity for objects and relations defined, object theory adds an axiom schema for substitution of identicals.

[3] The informal construction "In Peano Number Theory, $F0$" can be analyzed in object theory as well. A theory is analyzed as an abstract object that encodes propositions p by encoding the propositional properties of the form $[\lambda y\, p]$ (read this predicate as: being such that p). Then we can define "In theory T, $F0$" as $T[\lambda y\, F0]$, i.e., as T encodes the property *being such that 0 exemplifies F*. This analysis applies to any other mathematical individual κ, mathematical theory T, and constructions of the form "In theory T, $F\kappa$". For a full discussion of the analysis of mathematics within object theory, see [13].

In addition to the above principles, one other principle is added to the standard principles of second-order quantified modal logic, namely, the claim that if x possibly encodes a property F it necessarily encodes F:

$$\Diamond xF \rightarrow \Box xF$$

Thus, encoding predications are not relative to any circumstance; under the standard interpretation of the modal operators, this principle guarantees that if an encoding statement is true at any possible world, it is true at every possible world. By contrast, the truth of exemplification statements (and complex statements containing them) can vary from world to world.

2.2 Application to Possible Worlds

Object theory has been applied in a variety of ways. Our present focus is on Leibniz's theory of concepts, which includes a non-modal calculus of concepts, the concept containment theory of truth, and a modal metaphysics of concepts. However, to represent the latter, we must explain how possible worlds are analyzed in object theory. Though possible worlds (since [7]) are usually taken as semantically-primitive entities and used to formulate truth conditions for modal statements (as we did above, in explaining the axiom $\Diamond xF \rightarrow \Box xF$), object theory takes a different approach. Though the language of object theory includes modal operators, it uses these operators to *define* possible worlds as abstract objects of a certain kind and *derive* the main principles governing worlds from object-theoretic axioms. This, it is claimed, justifies the use of possible worlds when doing modal semantics (including in the modal semantics of object theory). So we will often contrast the possible worlds definable in object theory with the 'semantically primitive possible worlds' used in standard modal semantics that we sometimes reference. In what follows, we first rehearse the object-theoretic analysis of possible worlds and then rehearse the theory of Leibnizian concepts.

Possible worlds are defined as situations, where a *situation* is any abstract object that encodes only propositional properties of the form $[\lambda y\, p]$:

$$\mathit{Situation}(x) \ =_{df} \ A!x \,\&\, \forall F(xF \rightarrow \exists p(F = [\lambda y\, p]))$$

When x is a situation, we say that x *makes* proposition p *true* (or p is *true in* x), written $x \models p$, just in case x encodes *being such that* p:

$$x \models p \ =_{df} \ \mathit{Situation}(x) \;\&\; x[\lambda y\, p]$$

Then, we define a *possible world* to be any situation that *might* be such that it encodes all and only the true propositions [22]:

$$\mathit{World}(x) \ =_{df} \ \mathit{Situation}(x) \;\&\; \Diamond\forall p((x \models p) \equiv p)$$

If we then say that an object x is *maximal* just in case x is a situation and, for every proposition p, either p is true in x or $\neg p$ is true in x, i.e.,

$$\mathit{Maximal}(x) \ =_{df} \ \mathit{Situation}(x) \;\&\; \forall p((x \models p) \ \vee \ (x \models \neg p))$$

then it follows that every possible world is maximal:

$$\forall x(\mathit{World}(x) \rightarrow \mathit{Maximal}(x))$$

Moreover, let us say that an object x is *actual* just in case x is a situation such that every proposition true in x is true, i.e.,

$$\mathit{Actual}(x) \ =_{df} \ \mathit{Situation}(x) \;\&\; \forall p((x \models p) \rightarrow p)$$

It then follows that there is a unique actual world. That is, where $\exists! x\phi$ asserts that there is a unique x such that ϕ:

$$\exists! x(\mathit{World}(x) \;\&\; \mathit{Actual}(x))$$

The fundamental theorems of world theory are also provable, namely, that p is necessarily true if and only if p is true in all possible worlds, and p is possibly true if and only if p is true in some possible world [22]:

$$\Box p \equiv \forall x(\mathit{World}(x) \rightarrow x \models p)$$

$$\Diamond p \equiv \exists x(\mathit{World}(x) \;\&\; x \models p)$$

These theorems play an important role in the analysis of Leibniz's modal metaphysics of concepts, in which he asserts that if an object x exemplifies a property F but might not have, then not only does the *individual concept* of x *contain* the *general concept* of F, but there is a *counterpart* of the concept of x that doesn't contain the concept of F and that *appears at* some other possible world. One of our main goals is to represent and prove this claim within an automated reasoning environment.

2.3 Application to Leibniz's Theory of Concepts

As mentioned previously on several occasions, Leibniz's theory of concepts has three components: a non-modal calculus of concepts, the containment theory of truth, and a modal metaphysics of concepts. We can integrate and unify all three facets of Leibniz's work by deriving the main theorems of each within object theory. In the remainder of this section, we shall use the variables x, y, z as restricted variables that range just over abstract objects.

2.3.1 Leibniz's Non-modal Calculus of Concepts.

The first step of this integration is to recognize that abstract objects serve as a good analysis of Leibnizian concepts generally, so that we may define:

$$x \textit{ is a Leibnizian concept } (\text{'}C!x\text{'}) \ =_{df} \ A!x$$

Since Leibnizian concepts are abstract objects, we immediately obtain the first three theorems of Leibniz's [9], namely, that identity for concepts is reflexive, symmetric, and transitive. This is derivable from the definition of identity for abstract objects (see Section 2.1).[4]

The two final key definitions that yield, as theorems, the axioms of Leibniz's calculus of concepts [9] are: concept summation ($\oplus$) and concept inclusion ($\preceq$):

$$x{\oplus}y \ =_{df} \ \imath z(C!z \;\&\; \forall F(zF \equiv xF \vee yF))$$

$$x \preceq y \ =_{df} \ \forall F(xF \rightarrow yF)$$

From these two definitions, it follows that $\oplus$ is an idempotent, commutative, and associative operation on the concepts, and that $\preceq$ is a reflexive, anti-symmetric, and transitive condition [24].[5] Thus, if we think of concept summation as a *join* operation, Leibniz's 'calculus' of concepts is in effect a semi-lattice. Though we have not pursued the matter, the semi-lattice can be extended to a lattice by introducing a *meet* operation

[4] It is an easy logical theorem that $\Box\forall F(xF \equiv xF)$. But then, when x is a concept, it is abstract, and so it follows from the definition of identity that $x = x$. Using the principle of substitution of identicals, we can then derive the symmetry and transitivity of identity for abstract objects.

[5] Note we say *condition* here rather than *relation* because the definition of $\preceq$ has encoding subformulas and, as such, $\preceq$ is not guaranteed to be a relation.

$x \otimes y$, i.e., concept multiplication, by replacing the disjunction sign in the definition of $x \oplus y$ with an ampersand.

Here are some other key theorems derivable from the above theory of concepts (see [24], Theorems 25–27):

$$x \preceq y \;\equiv\; \exists z(x \oplus z = y)$$

$$x \preceq y \;\equiv\; x \oplus y = y$$

$$x \oplus y = y \;\equiv\; \exists z(x \oplus z = y)$$

Finally, Leibniz's notion of *concept containment* is just the converse of *concept inclusion*:

$$x \succeq y \;=_{df}\; y \preceq x$$

Thus, one can prove theorems analogous to the above that are stated in terms of concept containment instead of concept inclusion.

We have not yet brought automated reasoning tools to bear on the above object-theoretic *reconstruction* of Leibniz's algebra of concepts. Instead, the focus of our investigations was on the work described in the remainder of this section. However, in a separate project using Prover9, we verified *versions* of the above theorems in which $\oplus$ and $\preceq$ were taken as primitive and axiomatized instead of defined as in object theory.[6]

2.3.2 Leibniz's Containment Theory of Truth.

Though Leibnizian concepts are identified generally as abstract objects, special kinds of Leibnizian concepts can be defined. For example, there are general concepts of properties (e.g., the concept of being a king, etc.) and concepts of individuals (e.g., the concept of Alexander the Great). Both play a role in Leibniz's containment theory of truth.

First, we define "the concept of F" ('$\boldsymbol{c}_F$') as follows:

$$\boldsymbol{c}_F \;=_{df}\; \imath x(C!x \,\&\, \forall G(xG \equiv F \Rightarrow G))$$

In this definition, $F \Rightarrow G$ is defined as necessary implication: $\Box\forall x(Fx \rightarrow Gx)$. Thus, $\boldsymbol{c}_F$ is the concept that encodes exactly the properties that are necessarily implied by the property F.

Next, we define 'the concept of individual u' ('$\boldsymbol{c}_u$'), where u is a restricted variable ranging over ordinary individuals, as follows:

$$\boldsymbol{c}_u \;=_{df}\; \imath x(C!x \,\&\, \forall F(xF \equiv Fu))$$

For example, the concept of Alexander ($\boldsymbol{c}_a$) is the concept that encodes exactly the properties Alexander (a) exemplifies. Note that in this example, Alexander gets correlated with a concept that contains his properties. If we restate this using the concepts of simple set theory: the proper name 'a' is correlated with a set of properties. This recalls the treatment of proper names as generalized quantifiers [12].[7]

The definitions of $\boldsymbol{c}_F$ and $\boldsymbol{c}_u$ put us into a position to represent Leibniz's containment theory of truth. Leibniz asserts that a simple predication 'Alexander is king' ('Ka') is to be analyzed as: the concept Alexander contains the concept of being a king.[8] Where $\boldsymbol{c}_K$ is the concept of being a king and $\boldsymbol{c}_a$ is the concept of Alexander, we can represent Leibniz's analysis as:

$$\boldsymbol{c}_a \succeq \boldsymbol{c}_K$$

The equivalence of Ka and Leibniz's analysis $\boldsymbol{c}_a \succeq \boldsymbol{c}_K$ is *derivable* in object theory, since it is a general theorem that for any ordinary object u and property F:

Theorem 38, [24]:
$Fu \equiv \boldsymbol{c}_u \succeq \boldsymbol{c}_F$

[6] See `https://mally.stanford.edu/cm/leibniz/` for a description of this work and links to all the input and output files.

[7] Indeed, if we define 'the concept of every person' as the concept that encodes exactly the properties F such that every person exemplifies F, then the containment theory of truth described below will offer a unified 'subject-predicate' analysis of the sentences 'Alexander is happy' and 'Every person is happy'. On the containment theory of truth, the former is true because the concept of Alexander contains the concept of being happy, while the latter is true because the concept of every person contains the concept of being happy.

[8] Leibniz asserts his containment theory of truth in the following passage taken from the translation in [15], 18-19 (the source is [3], 51):

> ...every true universal affirmative categorical proposition simply shows some connection between predicate and subject (a *direct* connexion, which is what is always meant here). This connexion is, that the predicate is said to be in the subject, or to be contained in the subject; either absolutely and regarded in itself, or at any rate, in some instance; i.e., that the subject is said to contain the predicate in a stated fashion. This is to say that the concept of the subject, either in itself or with some addition, involves the concept of the predicate.

The translator titled the fragment from which this passage is taken as 'Elements of a Calculus'.

It is important to note that Theorem 38 is an example of a theorem that isn't a necessary truth. The reason is easy to see if we take possible worlds as semantically primitive and speak in terms of the classical semantics of modal logic: the formula Fu can change in truth value from world to world, but the formula $\boldsymbol{c}_u \succeq \boldsymbol{c}_F$ uses a term, $\boldsymbol{c}_u$, that is rigidly defined in terms of what is true at the actual world. $\boldsymbol{c}_u$ encodes all and only the properties that u in fact exemplifies. Hence, if u is in fact F (i.e., Fu is true at the actual world w_0) and there is a world w_1 where u fails to be F, then the left side of Theorem 38 is false at w_1 while the right side of Theorem 38 is true at w_1 (c_u is the object that encodes all the properties that u exemplifies at w_0, and since u does in fact exemplify F at w_0, one can show that c_u contains c_F).[9] The fact that the proof of Theorem 38 rests on a contingency actually works in Leibniz's favor: he introduced the notion of a hypothetical necessity in response to (his contemporary) Antoine Arnauld, who charged that he mistakenly analyzed the contingent claim Ka in terms of the necessary claim $\boldsymbol{c}_a \succeq \boldsymbol{c}_K$. Theorem 38 is indeed a kind of hypothetical necessity, if we understood that to mean that its proof depends on a contingency and that we can detach and then prove $\boldsymbol{c}_a \succeq \boldsymbol{c}_K$ only given the contingent Ka as a premise.

The above facts about Theorem 38 can be traced back to an interesting feature of object theory, namely, that rigid definite descriptions are governed by a logical axiom that fails to be a necessary truth.[10] When one includes such a rigidifying operator that is semantically interpreted with respect to the facts at the actual world of the model, then one must stipulate: the Rule of Necessitation may not be applied to any axiom governing the operator having the form of a conditional in which a (potentially contingent) formula ϕ appears on one side of the conditional in a non-rigid context and appears on the other side of the conditional in a rigid context. Nor can the Rule of Necessitation be allowed to apply to any theorem derived from such axioms. Note that the actuality operator is similar to the rigid definite description operator in this regard; for a full discussion of logical truths that aren't necessary, see [21]. As we shall see, this issue won't surface when we represent definite descriptions, since primitive descriptions will be eliminated under the standard Russellian analysis.

2.3.3 Leibniz's Modal Metaphysics of Concepts.

Finally, we reach the most compelling ideas in Leibniz's metaphysics, in which he uses the containment theory of truth to analyze modal predications and, in the course of doing so, relates concepts of individuals and possible worlds. In various passages, Leibniz talks about possible individuals.[11] However, it is generally thought that Leibniz's containment theory of truth was designed to replace talk of individuals having properties with talk of containment holding between concepts. Consequently, most commentators believe that we should interpret Leibniz's references to possible individuals as references to *concepts of individuals.* Leibniz does after all say that a concept of an individual may *appear* at a (unique) possible world.

To represent these ideas, we continue to use u as a restricted variable over ordinary individuals and use w as a restricted variable ranging over possible worlds (i.e., the worlds defined in an Section 2.2). We then define:

$$\mathit{RealizesAt}(u,x,w) \ =_{df} \ \forall F((w \models Fu) \ \equiv \ xF)$$

$$\mathit{AppearsAt}(x,w) \ =_{df} \ \exists u \mathit{RealizesAt}(u,x,w)$$

$$\mathit{IndividualConcept}(x) \ =_{df} \ \exists w \mathit{AppearsAt}(x,w)$$

From these definitions, it follows that every individual concept appears at a unique world ([24], Theorem 31):

$$\mathit{IndividualConcept}(x) \rightarrow \exists! w \mathit{AppearsAt}(x,w)$$

Moreover, not only is there a concept of the individual Alexander (which we've defined as $\boldsymbol{c}_a$), but for each possible world w, there is a concept of

[9] It is interesting to note that for some properties G, the proof that $\boldsymbol{c}_u$ encodes G will rest on a contingency (namely, when G is a property contingently exemplified by u), but the proof that c_F encodes G doesn't. That's because it is provable that if $F \Rightarrow G$ then $\Box(F \Rightarrow G)$.

[10] Instead of stating this logical axiom in its full generality, here is an example of an instance:

$$F\imath xGx \equiv \exists x(Gx \mathbin{\&} \forall y(Gy \rightarrow y\!=\!x) \mathbin{\&} Fx)$$

This is a version of Russell's analysis of definite description, first described in [18]. If the description $\imath xGx$ rigidly denotes the object that is uniquely G at the actual world (assuming there is one), then the above principle will fail to be necessarily true if there is a unique G at the actual world that is F at a world w_1, but where nothing is G at w_1 or where two distinct things are G.

[11] See, for example, the *Theodicy* ([10], 371 = [5], vi, 363), where he talks about the 'several Sextuses', and in a letter to Hessen-Rheinfels, where he talks about the 'many possible Adams' ([16], 51 = [5], ii, 20).

the individual of Alexander at w, $\boldsymbol{c}_a^w$, which encodes exactly the properties F that Alexander exemplifies at w. Thus, we may define:

$$\boldsymbol{c}_u^w \;=_{df}\; \imath x(C!x \;\&\; \forall F(xF \equiv w \models Fu))$$

Where $\boldsymbol{w}_\alpha$ is the actual world, it is easy to show that $\boldsymbol{c}_a$ is identical to $\boldsymbol{c}_a^{\boldsymbol{w}_\alpha}$ (i.e., the concept of Alexander is identical to the concept of Alexander at the actual world). Moreover, one can show that for any ordinary individual u, $\boldsymbol{c}_u$ is an individual concept, and that for any ordinary individual u and world w, $\boldsymbol{c}_u^w$ is an individual concept:

$$\forall u \mathit{IndividualConcept}(\boldsymbol{c}_u)$$

$$\forall u, w \mathit{IndividualConcept}(\boldsymbol{c}_u^w)$$

These facts put us in a position to see that both the Kripkean [8] and Lewisian [11] interpretation of possible objects can exist side-by-side (though Lewis's possible individuals are represented at the level of concepts). Kripke believes that a modal claim such as "Obama might have had a son" is true because

> there is a possible world where *Obama himself* has a son.

By contrast, Lewis takes this modal claim to be true because

> there is a possible world where *a counterpart of Obama* has a son.

The precise Leibnizian picture we've developed enables us to show how Kripke's view holds with respect to ordinary individuals, while Lewis's view holds with respect to Leibnizian complete individual concepts.

To see that Kripke's view holds with respect to ordinary individuals, we need only observe that it is Obama himself who fails to have a son in our world but who has a son at some other world. However, to see why Lewis's view holds with respect to concepts of ordinary individuals, we must first partition the concepts of ordinary individuals into groups of counterparts. Let (italic, non-bold) c, c' range over individual concepts. Then we may define:

$$\mathit{Counterparts}(c, c') =_{df} \exists u \exists w_1 \exists w_2 (c = \boldsymbol{c}_u^{w_1} \;\&\; c' = \boldsymbol{c}_u^{w_2})$$

In other words, individual concepts c and c' are counterparts whenever there is an ordinary object u and worlds w_1 and w_2 such that c is the concept of u-at-w_1 and c' is the concept of u-at-w_2. So, if w' is a world where Obama does have a son, $\boldsymbol{c}_o^{w'}$ is a counterpart of the concept of Obama ($\boldsymbol{c}_o$), given that $\boldsymbol{c}_o = \boldsymbol{c}_o^{\boldsymbol{w}_\alpha}$. Obama, $\boldsymbol{w}_\alpha$ and w' are thus witnesses to the definition of $\mathit{Counterparts}(\boldsymbol{c}_o, \boldsymbol{c}_o^{w'})$.

Now, as we saw above, individual concepts appear at a unique world. So they are, in some sense, world-bound individuals. Thus, we obtain Lewis-style truth conditions for modal claims in the domain of Leibnizian individual concepts, given the following *fundamental theorem* (applied to Alexander):

> If Alexander is king but might not have been, then:
>
> (a) the concept of Alexander contains the concept of being a king, and
>
> (b) some individual concept that is a counterpart to the concept of Alexander fails to contain the concept of being a king and *appears at* some non-actual possible world.

Formally and generally, where u is any ordinary object, c is any individual concept, and F is any property, we have:

Theorem 40a [24]:
$$(Fu \;\&\; \Diamond\neg Fu) \to [\boldsymbol{c}_u \succeq \boldsymbol{c}_F \;\&\; \exists c(\mathit{Counterparts}(c, \boldsymbol{c}_u) \;\&\; c \not\succeq \boldsymbol{c}_F \;\&\; \exists w(w \neq \boldsymbol{w}_\alpha \;\&\; \mathit{Appears}(c, w)))]$$

Similarly, we have:

> If Obama doesn't have a son but might have, then:
>
> (a) the concept of Obama fails to contains the concept of having a son, and
>
> (b) some individual concept that is a counterpart to the concept of Obama contains the concept of having a son and *appears at* some non-actual possible world.

Formally and more generally, this becomes:

Theorem 40b [24]:
$$(\neg Fu \;\&\; \Diamond Fu) \to [\boldsymbol{c}_u \not\succeq \boldsymbol{c}_F \;\&\; \exists c(\mathit{Counterparts}(c, \boldsymbol{c}_u) \;\&\; c \succeq \boldsymbol{c}_F \;\&\; \exists w(w \neq \boldsymbol{w}_\alpha \;\&\; \mathit{Appears}(c, w)))]$$

The main goal of our efforts to implement Leibniz's modal metaphysics computationally was to obtain a proof of the above two theorems using an automated reasoning engine. In what follows, we'll work our way to an understanding of our computational implementation of Theorem 40a.

3 Summary of Our Representational Techniques

The fundamental idea behind our work in implementing object theory in an automated reasoning environment is this: instead of building a customized theorem prover that understands the syntax of object theory, we use the language of standard theorem provers to represent the *first-order truth conditions* of the statements of object theory. The first order truth conditions can be understood in terms of the object theory's natural semantics and model theory. This is entirely appropriate because the minimal models of object theory reveal that despite its second-order syntax, it has general Henkin models [6]. We did not employ a mechanical procedure for translating the formulas of object theory into TPTP syntax, but rather carried it out by hand. In constructing the translations, we adopted various conventions, some of which are discussed below.

Our basic convention was to translate the second-order quantified modal syntax of object theory into $\mathsf{FOL}_{=}$, supplemented with predicates that sort individuals into four domains: objects, properties, propositions, and points. Our representation can be written directly in TPTP syntax without further processing. In what follows, we summarize the techniques we developed in order to produce TPTP problem files for the theorems in [24].

3.1 Representing Second-Order Syntax Using First-Order Syntax

The most important first step of the process is to recognize that in the semantics of object theory, 1-place properties have an exemplification extension among objects and that this extension varies from possible world to possible world. However, since possible worlds are going to be one of the targets of object-theoretic analysis, we call the semantically-primitive possible worlds *points*. Moreover, we refer to 0-place relations as *propositions*. Thus to translate modal claims involving the individual variables $x, y, z, \ldots$, property variables $F^1, G^1, \ldots$ and propositional variables $F^0, G^0, \ldots$ (which we write using $P, Q, \ldots$), we introduce the following basic sorts:

```
object(X)
property(F)
proposition(P)
point(D)
```

Moreover, since the simple and complex predications in object theory take place in a modal language, we adopted the convention of introducing an extra argument place in primitive or defined conditions, which relativizes them with respect to a point `D`. When translating explicitly modal claims, that extra argument place can be bound by a quantifier over points.

Whereas uppercase `D` is a variable ranging over points, we represent a basic (non-modal) predication of the form Fx using a named point:

```
ex1_wrt(F,X,d)
```

Here, `d` is the semantically primitive 'actual world' that serves as the distinguished element of the domain of possible worlds found in classical modal semantics. In general, then, since a formula like Fx can appear within a modal context, we represent it with the primitive condition `ex1_wrt(F,X,D)`, which has an argument place for point `D`. Note that we also add as an axiom the *right-handed sorting* rule that asserts that if `ex1_wrt(F,X,D)`, then `F` is a property, `X` is an object, and `D` is a point:

```
fof(sort_ex1_wrt,type,
 (!  [F,X,D] : (ex1_wrt(F,X,D) =>
  (property(F) & object(X) & point(D))))).
```

This, in effect, tells the theorem prover that we're primarily interested in models in which the arguments of the relation `ex1_wrt` are entities of the appropriate sorts. Such right-handed sorting rules allow us to represent facts that come for free in a second-order language.

Next, we represent a basic modal predication of the form $\Box Fx$ as:

```
(!  [D] : (point(D) => ex1_wrt(F,X,D)))
```

Possibility claims are represented in a similar way, using existential quantifications over points.

To represent the primitive predicate '$E!x$' and the defined predicates '$O!x$' and '$A!x$', we introduced the property constants `e`, `o` and `a`. Just as in object theory, `e` is primitive. But `o` and `a` are defined as (cf. the definitions described in Section 2.1):

```
fof(o,definition,
 (!  [X,D] : ((object(X) & point(D)) => (ex1_wrt(o,X,D) <=>
  (?  [D2] :  (point(D2) & ex1_wrt(e,X,D2))))))).
```

```
fof(a,definition,
 (!  [X,D] : ((object(X) & point(D)) => (ex1_wrt(a,X,D) <=>
  ~(?  [D2] :  (point(D2) & ex1_wrt(e,X,D2))))))).
```

Now to introduce the constant `c` to denote the property of *being a concept*, we asserted that the property of being a concept is identical to the property of being abstract, i.e.,

```
fof(being_a_concept_is_being_abstract,axiom,c=a).
```

The above techniques are an important first step towards solving the problem of representing the object-theoretic definitions of the various metaphysical kinds used by Leibniz, such as concepts of individuals, concepts of properties, possible worlds, etc.

3.2 Representing the Two Modes of Predication

Now the distinguishing feature of the language of object theory is that it has two fundamental modes of predication. In addition to predications of the form Fx (familiar from standard FOL), there are also predications of the form xF. We represent the latter as `enc_wrt(X,F,D)`, and require the following right-handed sorting rule:

```
fof(sort_enc_wrt,type,
 (!  [X,F,D] : (enc_wrt(X,F,D) =>
  (object(X) & property(F) & point(D))))).
```

The formula `enc_wrt(X,F,D)` will appear in several of the definitions that are given below.

3.3 Representing Identity Claims

Recall the disjunctive definition of object-theoretic identity $x = y$ in Section 2.1. To represent that definition, we developed two preliminary definitions, one for the identity of ordinary objects (`o_equal_wrt`) and one for the identity of abstract objects (`a_equal_wrt`).[12] We then represented $x = y$ in terms of the general notion `object_equal_wrt(X,Y,D)` by stipulating that `object_equal_wrt(X,Y,D)` holds if and only if either `o_equal_wrt(X,Y,D)` or `a_equal_wrt(X,Y,D)`, as follows:

```
fof(object_equal_wrt,definition,
 (!  [X,Y,D] : ((object(X) & object(Y) & point(D)) =>
  (object_equal_wrt(X,Y,D) <=>
   (o_equal_wrt(X,Y,D) | a_equal_wrt(X,Y,D)))))).
```

Finally, we then connected general identity `object_equal_wrt(X,Y,D)` with the built-in equality of the reasoning system. This *bridge principle* asserts:

```
fof(object_equal_wrt_implies_identity,theorem,
 (!  [X,Y] : ((object(X) & object(Y)) =>
  (?  [D] : (point(D) & object_equal_wrt(X,Y,D)) =>
   X = Y)))).
```

That is, if objects `X` and `Y` are `object_equal` at some point `D`, then they are identical. This definition suffices because it is a theorem that any objects `o_equal` at some point are `o_equal` at every point, and also a theorem that any objects `a_equal` at some point are `a_equal` at every point.

With the above bridge principle in place, inferences about object-theoretic identity can be drawn by the automated reasoning engine using system equality (e.g., via demodulation).

3.4 Representing Definite Descriptions

Here are two examples of how we represent definite descriptions. Recall that we introduced the term $\boldsymbol{c}_u$ to denote *the* concept c that encodes all and only the properties that u in fact exemplifies, and we introduced the term $\boldsymbol{c}_F$ to denote *the* concept c that encodes all and only the properties necessarily implied by F. Consider first how we represent $\boldsymbol{c}_u$. We begin by introducing the relational condition `concept_of_individual_wrt(X,U,D)`, which holds just in case `U` is an ordinary object and `X` is a concept (i.e.,

[12] The definition of `o_equal_wrt` is:

```
fof(o_equal_wrt,definition,
 (!  [X,Y,D] : ((object(X) & object(Y) & point(D)) =>
  (o_equal_wrt(X,Y,D) <=> (ex1_wrt(o,X,D) & ex1_wrt(o,Y,D) &
   (!  [D2] :  (point(D2) => (!  [F] : (property(F) =>
        (ex1_wrt(F,X,D2) <=> ex1_wrt(F,Y,D2)))))))))).
```

This says that `X` and `Y` are `o_equal` with respect to point `D` just in case `X` and `Y` are both ordinary objects and at every point `D2`, they exemplify the same properties. A similar definition defines: `X` and `Y` are `a_equal` with respect to point `D` just in case `X` and `Y` are both abstract objects and at every point `D2`, they encode the same properties.

abstract object) that encodes exactly the properties `F` such that `U` exemplifies `F`:

```
fof(concept_of_individual_wrt,definition,
 (! [X,U,D] : ((object(X) & object(U) & point(D)) =>
  (concept_of_individual_wrt(X,U,D) <=> (ex1_wrt(c,X,D) &
   ex1_wrt(o,U,D) & (! [F] : (property(F) =>
    (enc_wrt(X,F,D) <=> ex1_wrt(F,U,D))))))))).
```

We then introduce `is_the_concept_of_individual_wrt(X,U,D)` as holding whenever `X` is a concept of individual `U` with respect to point `D` and anything `Z` that is a concept of individual `U` with respect to point `D` is `object_equal` to `X`:

```
fof(is_the_concept_of_individual_wrt,definition,
 (! [X,U,D] : ((object(X) & object(U) & point(D)) =>
  (is_the_concept_of_individual_wrt(X,U,D) <=>
   (concept_of_individual_wrt(X,U,D) &
    (! [Z] : (concept_of_individual_wrt(Z,U,D) =>
     object_equal_wrt(Z,X,D)))))))).
```

Here `X` corresponds to $\boldsymbol{c}_u$ when `is_the_concept_of_individual_wrt(X,U,d)`.

Consider second how we represent $\boldsymbol{c}_F$. We begin by introducing the relational condition `concept_of_wrt(Y,F,D)`, which holds just in case `Y` is a concept that encodes just the properties necessarily implied by F. Formally:

```
fof(concept_of_wrt,definition,
 (! [Y,F,D] : ((object(Y) & property(F) & point(D)) =>
  (concept_of_wrt(Y,F,D) <=>
   (ex1_wrt(c,Y,D) & (! [G] : (property(G) =>
    (enc_wrt(Y,G,D) <=> implies_wrt(F,G,D))))))))).
```

Then we introduce `is_the_concept_of_wrt(Y,F,D)` as holding whenever `Y` is a concept of property `F` at point `D` and anything `Z` that is a concept of `F` at `D` is `object_equal` to `Y`:

```
fof(is_the_concept_of_wrt,definition,
 (! [Y,F,D] : ((object(Y) & property(F) & point(D)) =>
  (is_the_concept_of_wrt(Y,F,D) <=>
   (concept_of_wrt(Y,F,D) & (! [Z] : (object(Z) =>
    (concept_of_wrt(Z,F,D) => object_equal_wrt(Z,Y,D))))))))).
```

Thus, `Y` corresponds to $\boldsymbol{c}_F$ when `is_the_concept_of_wrt(Y,F,d)`.[13] All of the above definitions play an important role in the statement and proof of the fundamental theorem of Leibniz's modal theory of concepts.

3.5 Representing λ-Expressions

As an example of how we represented λ-expressions, consider $[\lambda z\, Py]$, which denotes the property: being a z such that y exemplifies P. We discussed such λ-expressions in footnote 3 and at the beginning of Section 2.2 (on world theory). To properly understand these expressions, note that, in object theory, it is a theorem that for every property P and object y, there exists a proposition Py. Moreover, for each such proposition, object theory's comprehension principle for properties asserts (cf. footnote 2):

$$\exists F \Box \forall x(Fx \equiv Py)$$

i.e., there is a property F such that, necessarily, an object x exemplifies F if and only if Py. We use the λ-expression $[\lambda z\, Py]$ to denote such a property. It obeys the principle:

$$\Box([\lambda z\, Py]x \equiv (Py)^x_z)$$

However, the variable z bound by the λ in $[\lambda z\, Py]$ is vacuously bound since it doesn't appear in Py. So $(Py)^x_z$ (i.e., the result of substituting x for z in Py) is just the formula Py. Hence we have: $\Box([\lambda z\, Py]x \equiv Py)$, i.e., necessarily, x exemplifies *being such that Py* if and only if Py.

We represented these facts as follows:

```
fof(existence_proposition_plug1,axiom,
 (! [X,F] : ((object(X) & property(F)) =>
  (? [P] : (proposition(P) & plug1(P,F,X)))))).

fof(proposition_plug1_truth,definition,
 (! [X,F,P] : ((object(X) & property(F) & proposition(P)) =>
  (plug1(P,F,X) => (! [D] : (point(D) =>
  (true_wrt(P,D) <=> ex1_wrt(F,X,D)))))))).

fof(existence_vac,axiom,
 (! [P] : (proposition(P) =>
   (? [Q] : (property(Q) & is_being_such_that(Q,P)))))).
```

[13] It is important to note here that, in contrast to the concept of an individual, we need not have linked `Y` to the concept of `F` at the distinguished point `d`, given what we said in footnote 9.

```
fof(truth_wrt_vac,axiom,
 (!  [P,Q] : ((proposition(P) & property(Q)) =>
  (is_being_such_that(Q,P) =>
   (!  [D,X] : ((point(D) & object(X)) =>
    (ex1_wrt(Q,X,D) <=> true_wrt(P,D)))))))).
```

The first asserts that for any property F and object X, there is a proposition P obtained by *plugging* X into F, where the truth conditions for plugging are defined by the second principle as: if P is the proposition obtained by plugging X into F, then for every point D, P is true with respect to D whenever X exemplifies F with respect to D. The third asserts that for every proposition P, there is a property Q such that Q is being such that P. The fourth asserts that if Q is being such that P, then for any point D and object X, X exemplifies Q at D if and only if P is true at D.

4 Representing the Fundamental Theorem

We now apply the techniques just summarized to the representation of one of the two fundamental theorems described at the end of Section 2, namely, Theorem 40a. The antecedent of Theorem 40a is:

$$Fu \,\&\, \Diamond\neg Fu \qquad \text{(A)}$$

Since the variable 'u' is a restricted variable ranging over ordinary objects, we represent the first conjunct as:

```
ex1_wrt(o,U,d) & ex1_wrt(F,U,d),
```

where o is the property of being ordinary, F is a variable ranging over properties, U is a variable ranging over objects, and d is the distinguished point. By using sortal predicates to make everything explicit, the conjunction (A) can be represented as follows:

```
object(U) & property(F) & ex1_wrt(o,U,d) & ex1_wrt(F,U,d) &
 (?  [D] : (point(D) & ~ex1_wrt(F,U,D)))
```

$(\overline{\mathrm{A}})$

In other words, the antecedent of Theorem 40a becomes:

> (If) U is an object, F is a property, U exemplifies being ordinary at d, U exemplifies F at d, and there is a point D such that U fails to exemplify F at D, (then) ...

Now the first conjunct of the consequent of Theorem 40a is:

$$\boldsymbol{c}_u \succeq \boldsymbol{c}_F \qquad \text{(B)}$$

This is not a theorem of object theory, though it follows from the facts that $Fu \equiv \boldsymbol{c}_u \succeq \boldsymbol{c}_F$ (referenced earlier as Theorem 38a) and the premise Fu. Hence it follows from the antecedent of Theorem 40a. If we recall the earlier definitions of $\boldsymbol{c}_u$, $\boldsymbol{c}_F$ and $\succeq$, we can represent this clause as follows, in which $\boldsymbol{c}_u$ is represented by X, $\boldsymbol{c}_F$ is represented by Y and $\succeq$ is represented by contains_wrt(X,Y,d):

```
(?  [X,Y] : object(X) & object(Y) & ex1_wrt(c,X,d) &
 ex1_wrt(c,Y,d) & is_the_concept_of_individual_wrt(X,U,d) &
  is_the_concept_of_wrt(Y,F,d) & contains_wrt(X,Y,d))
```

$(\overline{\mathrm{B}})$

The first four conjuncts of $(\overline{\mathrm{B}})$ tell us that X and Y are both objects and, in particular, both concepts. The fifth and sixth conjuncts of $(\overline{\mathrm{B}})$, i.e.,

```
is_the_concept_of_individual_wrt(X,U,d)

is_the_concept_of_wrt(Y,F,d)
```

were defined in Section 3.4. We therefore know that the X and Y asserted to exist in $(\overline{\mathrm{B}})$ corresponds to $\boldsymbol{c}_u$ and $\boldsymbol{c}_F$, respectively, in the language of object theory.

The final conjunct of $(\overline{\mathrm{B}})$ is contains_wrt(X,Y,d). This asserts that object X contains object Y at point d, where this is defined this as:

```
fof(contains_wrt,definition,
 (!  [X,Y,D] : ((object(X) & object(Y) & point(D)) =>
  (contains_wrt(Y,X,D) <=> included_in_wrt(X,Y,D))))).
```

Here, included_in_wrt(X,Y,D) is defined as you might expect given our discussion of $\succeq$ in Section 2.3.[14]

Finally, we turn to the second conjunct of the consequent of Theorem 40a. It asserts:

[14] The definition is:

```
fof(included_in_wrt,definition,
 (!  [X,Y,D] : ((object(X) & object(Y) & point(D)) =>
  (included_in_wrt(X,Y,D) <=>
   (ex1_wrt(c,X,D) & ex1_wrt(c,Y,D) &
    (!  [F] : (property(F) => (enc_wrt(X,F,D) => enc_wrt(Y,F,D))))))))).
```

$$\exists c(\mathit{Counterparts}(c,\boldsymbol{c}_u) \,\&\, c \not\succeq \boldsymbol{c}_F \,\&\, \exists w(w \neq \boldsymbol{w}_\alpha \,\&\, \mathit{Appears}(c,w)))$$

Given that we already know X represents $\boldsymbol{c}_u$ and Y represents $\boldsymbol{c}_F$, this becomes represented as follows:

```
(? [Z] : (object(Z) & ex1_wrt(c,Z,d) & counterparts_wrt(Z,X,d) &
 ~contains_wrt(Z,Y,d) & (?  [A,W] : (object(A) & object(W) &
  is_the_actual_world_wrt(A,d) & world_wrt(W,d) &
   ~equal_wrt(W,A,d) & appears_in_wrt(Z,W,d))))).
```

We can now represent Theorem40a:

```
fof(theorem_40a,conjecture,
 (!  [U,F] : ((object(U) & property(F)) =>
  ((ex1_wrt(o,U,d) & ex1_wrt(F,U,d) &
   (?  [D] : (point(D) & ~ex1_wrt(F,U,D)))) =>
    (?  [X,Y] : (object(X) & object(Y) & ex1_wrt(c,X,d) &
     ex1_wrt(c,Y,d) & is_the_concept_of_individual_wrt(X,U,d) &
      is_the_concept_of_wrt(Y,F,d) & contains_wrt(X,Y,d) &
       (?  [Z] : (object(Z) & ex1_wrt(c,Z,d) &
        counterparts_wrt(Z,X,d) & ~contains_wrt(Z,Y,d) &
         (?  [A,W] : (object(A) & object(W) &
          is_the_actual_world_wrt(A,d) & world_wrt(W,d) &
           ~equal_wrt(W,A,d) & appears_in_wrt(Z,W,d)))))))))))).
```

Theorem 40b has a similar representation. The problem files for both Theorem40a and Theorem40b are available online.[15] A web page containing the links to all the relevant files is also available.[16]

5 Techniques for Speeding Up the Workflow

Our work consisted of developing a theory (i.e., a structured collection of theorems and definitions) in TPTP notation. Faced with the task of adding premises to a TPTP file for some conjecture, we generally used [24] as a guide. To formalize (proofs of) theorems, we proceeded in the usual naive way. If the original proof referred to a previous theorem, we included it in the TPTP file. Similarly, whenever defined notions appeared in a conjecture or premise, we included their definitions in the TPTP file. Our workflow can intuitively be understood as taking a kind of poor man's closure operation: first we looked at the primitive and defined notions that appeared in the conjecture to be proved and then we kept adding to the file those axioms, definitions, and previous theorems governing the primitive and defined notions that we thought would be needed to yield a proof of the conjecture.

To facilitate constructing this "closure", we wrote a script that inspects a TPTP file and reports on the predicate symbols and function symbols appearing in it. (The script is essentially just a front end to the standard GetSymbols tool distributed with TPTP.) The procedure is embarrassingly simple, but it prevented us from wasting time trying to diagnose a countersatisfiable conjecture that failed because of lack of information about one of the notions in the conjecture. The script thus highlighted in red those symbols that occur exactly once in the problem (i.e., *hapax legomena*). This is a quick check for whether there is a gap in the problem because, intuitively, a predicate or function that occurs exactly once is a red flag. Of course, such a heuristic is far from complete. In certain situations, a problem may well be solvable while having symbols that appear exactly once. Equality counts as an undefined binary predicate symbol from our program's point of view, and at times we worked with problems where a single equation was present in the problem. At other times, it is acceptable for there to be primitive (undefined) notions that by chance do occur exactly once.

Another useful program we found ourselves in need of and developed was a tool for running multiple theorem provers and extracting the sets of premises used in the proofs (TSTP/TPTP derivations). It is quite interesting to see that different theorem provers "react" differently to one and the same theorem proving problem. After relying on the theorem provers to tell us which premises they used in a proof, we systematically tried removing premises and testing for the existence of a proof or a countermodel using the reduced set of premises. What we found is that our premise sets were almost always bigger than necessary. At times, a surprisingly large number of premises could be cut. We were often delightfully puzzled into rethinking our initial proof because we had expected that certain lemmas or definitions could not be removed. In order to be very clear about the power of the axioms, we wanted the extra insight that came from minimizing the premise sets.

The above tools, developed to prune unneeded concepts and premises

[15] See `https://mally.stanford.edu/cm/concepts/theorem40a.p` and `theorem40b.p`.

[16] See `https://mally.stanford.edu/cm/concepts/`.

from the proof of a given conjecture, were bundled together to make the Tipi program, which is written in Perl and available online.[17] Tipi was constructed as a catch-all tool for our formalization project. Tipi has proved of value in other formalization projects as well [1,2].

When we first started our project, we produced a separate input file for each theorem in [24]. This works fine as long as one doesn't end up having to go back and redo theorems when representational improvements are discovered. In the course of working on theorem-proving problems, we often had to make on-the-fly adjustments to our representations. If one regards our theory as a kind of tree of dependencies, in which formulas depend (either by definition or by derivation) on other formulas, such on-the-fly changes can quickly lead to confusion. One may confidently think that a small change to a formula ϕ makes no difference to the provability of a theorem ψ that depends it, only to be shown wrong by a countermodel. In general, *any* change to a formula reopens the question of whether some other dependent formula is provable; indeed, any change to a prior axiom, definition, or theorem, requires checking all the theorems that depended on that changed formula. Once dozens of theorems are involved, one has to ensure that axioms, definitions, and theorems are kept synchronized across multiple files. We sometimes were satisfied that a problem was solved only to have to revisit the problem when we discovered we could correct or improve the formulation some principle.

This problem of dependence is of course not unique to theorem proving; it is clearly an old, well-recognized issue in software engineering generally. In our case, we designed a suite of makefiles to help keep ourselves honest about the status of our theory as we made changes to it. The solution we arrived at is to regenerate problems from a master file. Each formula capable of generating a problem, that is, each theorem or lemma, is annotated with the axioms, definitions, sorts, theorems, and lemmas that are used in its derivation. The TPTP problem files are generated automatically from a master file containing the latest version of the dependencies.

[17] https://github.com/jessealama/tipi and http://arxiv.org/abs/1204.0901.

6 Observations

6.1 What We've Learned

Although our computational study of [24] didn't reveal any errors of reasoning, we did come away from the research with some new insights about the implementation of object theory using automated reasoning tools. One of the interesting things we learned concerned the demands that our representational methods placed on the definition of notions from object theory. Originally, we thought that it might help cut down the search space for proofs if we prefaced each definiendum with an antecedent that both sorted the variables and also introduced any restrictions on the variables. After all, reasoning with restricted variables eliminates inference steps and thus potential errors of reasoning. So, for example, situations are definable in object theory as abstract objects of a certain kind. We wondered whether proof search would be more efficient with this restriction, i.e., if we defined the world-relative condition `situation_wrt(X,D)` only for those objects `X` known to be abstract, as follows:

```
fof(situation_wrt,definition,
 (!  [X,D] : ((object(X) & point(D)) => (ex1_wrt(a,X,D) =>
  (situation_wrt(X,D) <=> (!  [F] : (property(F) => (enc_wrt(X,F,D) =>
   (?  [P] : (proposition(P) & is_being_such_that(F,P))))))))))).
```

However, in the end, we discovered that our provers do better if we don't use restricted variables when introducing the definiendum. The more general way of formulating such definitions is to introduce the definiendum as soon as the variables in the argument places are sorted. On that method, the above definition becomes:

```
fof(situation_wrt,definition,
 (!  [X,D] : ((object(X) & point(D)) => (situation_wrt(X,D) <=>
  (ex1_wrt(a,X,D) & (!  [F] : (property(F) => (enc_wrt(X,F,D) =>
   (?  [P] : (proposition(P) & is_being_such_that(F,P))))))))))).
```

Thus, we adhered to the following format for introducing an n-place condition `Definiendum(X1,...,Xn)`:

```
(!  [X1,...,Xn]:  ((sort1(X1) & ...  & sortn(Xn)) =>
 (Definiendum(X1,...,Xn) <=> ...X1...Xn...))).
```

Another interesting question that arose was when to formulate our theorems in their most general modal form, i.e., as necessary truths (prefaced by a universal quantifier over all points), as opposed to formulating them as non-modal facts that hold just of the distinguished point d. In many modal systems, this question doesn't arise since every theorem is a necessary truth. But object theory allows for reasoning with contingent premises and with a contingent axiom governing rigid definite descriptions. We noted earlier that the presence of rigid definite descriptions in certain contexts is a tip-off that it may be inappropriate to use the Rule of Necessitation (see especially the discussion following Theorem 38). One has to keep track of theorems that are proved with a contingent premise or that depend on the contingent axiom governing rigid definite descriptions.

In Section 2.3, we introduced Theorem 38, which asserts that an ordinary object u exemplifies F if and only if the concept of individual u contains the concept of property F. This theorem is a key part of the proof of Theorem 40a. It is important to recognize that Theorem 38 should be proved only in the following form (as a fact about the distinguished point d):

```
fof(theorem38,theorem,
 (!  [U,F] : ((object(U) & property(F)) => (ex1_wrt(o,U,d) =>
  (?  [Y,Z] : (object(Y) & object(Z) & ex1_wrt(c,Y,d) &
   ex1_wrt(c,Z,d) & is_the_concept_of_individual_wrt(Y,U,d) &
    is_the_concept_of_wrt(Z,F,d) &
     (ex1_wrt(F,U,d) <=> contains_wrt(Y,Z,d)))))))).
```

and *not* in the following form (as a fact about every point D):

```
fof(theorem38,theorem,
 (!  [U,F] : ((object(U) & property(F)) => (!  [D] : (point(D) =>
  (ex1_wrt(o,U,D) => (?  [Y,Z] : (object(Y) & object(Z) &
   ex1_wrt(c,Y,D) & ex1_wrt(c,Z,D) &
    is_the_concept_of_individual_wrt(Y,U,D) &
     is_the_concept_of_wrt(Z,F,D) &
      (ex1_wrt(F,U,D) <=> contains_wrt(Y,Z,D)))))))))).
```

When representing object theoretic claims, one always has to ask: is this provably true only with respect to the distinguished point d or is it provable for every point D? Of course, one must take care not to get confused by the fact that *possible worlds* are defined in object theory, and so we can express claims that have both variables ranging over defined possible worlds as well as variables ranging over the primitive sort point. The question of when to represent a theorem as a necessary truth affects only those claims involving the modal operator 'necessarily', not claims about possible worlds *per se*.[18]

An interesting point emerged about representing object theory's two main axiom schemata. Basically, we adopted the expedient of representing only the instances of the schemata that we needed as a premise to prove a conjecture. For example, as noted earlier, the main comprehension schema for abstract objects asserts:

$$\exists x(A!x \,\&\, \forall F(xF \equiv \phi)), \text{ provided } x \text{ doesn't occur free in } \phi$$

Since there is no way to represent schemata in first-order syntax, our policy was this: if any theorem that required the *existence* of an abstract object given by some instance of the above schema, then we formulated the particular instance as a premise. So, for example, if a theorem required the existence of the concept of individual u, we would represent the following instance:

$$\exists x(A!x \,\&\, \forall F(xF \equiv Fu))$$

Then, given the definition of the concept of individual u, we would be assured that the domain contained such a concept.

We also followed this procedure to address the problem of representing the β-conversion schema. Since we don't have a general way of representing all the various different λ-abstracts in $\mathsf{FOL}_=$, we simply had to manually represent various λ-abstracts and axiomatize them as needed. Thankfully, few instances were needed to complete the formalization.

We see two possible ways to represent these two axiom schemata in full generality. One is to reason syntactically about the formulas allowed

[18] In general, we adopted the policy of proving necessitations of theorems only when they were required for the proof of another theorem. For example, the following necessary truth was needed for the proof of Theorem 40a:

```
fof(uniqueness_of_concept_of_individual_in_wrt,lemma,
 (! [D] : (point(D) => (![X,Y,U,W] : ((object(X) & object(Y) &
  object(U) & object(W)) => (world_wrt(W,D) =>
   ((concept_of_individual_in_wrt(X,U,W,D) &
    concept_of_individual_in_wrt(Y,U,W,D)) =>
     a_equal_wrt(X,Y,D)))))))).
```

This asserts that for every point D, if X and Y are both concepts of the individual U with respect to D, then X and Y are identical abstract objects with respect to D.

in instances of the schema. The other is to formulate them in third-order logic, analogously to the way in which the induction axiom for arithmetic can be formulated as a single second-order axiom rather than as a schema for generating first-order axioms.

6.2 Future Work

At the time of writing, we haven't yet proved every lemma upon which Theorems 40a and 40b depend; only a few remain. Once the work is complete, we hope to put the theorems into a form that can be submitted to the TPTP Library. This will require an additional step of determining which of the axioms, definitions, lemmas, sorting principles, etc., constitute the core part of the theory. Once we identify the core part of the theory, we can look for a model of all of the key principles upon which Theorems 40a and 40b depend. We've found models of the premise sets in each of the separate input files for the theorems we've proved thus far, but until the work is complete, we won't be in a position to identify the core group of principles that require a consistency check. As of release 6.1.0 of the TPTP Library, there is a new section devoted to philosophy. We plan to submit this theory for inclusion in that section.

One long-term goal of our project is to identify desiderata for the design and implementation of a customized, native prover for object theory. A customized prover would allow us to input formulas that more closely resemble those of object theory. Furthermore, customized theorem provers and model builders might be able to recognize subformulas, recognize which formulas have no encoding subformulas, generate instances of the comprehension schema for relations, and generally be more attuned to the special features of object theory. Reasoning in object theory is more structured than simply throwing a set of formulas at a theorem prover and looking for a refutation. There are dependencies such as the dependence of definitions on their justifying theorems, and the restriction on the Rule of Necessitation to formulas that do not depend on contingent assumptions. This makes the definition of provability in object theory more subtle, which understandably complicates the implementation of any system that tries to be faithful to it.

However, there may be obstacles to developing a native theorem prover for object theory. If the work in [14] is correct, there is a feature of object theory that suggests it may be difficult to adapt those existing reasoning engines which are based on some form of functional type theory. For the discussion in [14] established that object theory (a) contains formulas that neither are terms themselves nor can be converted to terms by λ-abstraction, and therefore (b) involves reasoning that seems to be capturable only in the logic of relational rather than functional type theory. Consequently, if existing automated reasoning engines depend essentially on some form of functional type theory to define and navigate the search space for finding proofs, then it may be that new methods (e.g., ones that work in a relational type-theoretic environment and not just in a functional type-theoretic environment) will have to be incorporated into the design and implementation of a customized prover for object theory.

Finally, if new methods and tools are developed to make the process go more quickly and smoothly, it should be easier to investigate object theory's Frege-style derivation of the Dedekind-Peano axioms for arithmetic [23].

References

[1] Alama, J., 'Complete independence of an axiom system for central translations', *Note di Matematica*, 33 (2013): 133–142.

[2] Alama, J., 'The simplest axiom system for hyperbolic geometry revisited, again,' *Studia Logica*, 102(3) (2014): 609–615.

[3] Couturat, L. (ed.), *Opuscules et fragments inédits de Leibniz*, Paris: F. Alcan, 1903.

[4] Fitelson, B., and E. Zalta, 'Steps toward a computational metaphysics', *Journal of Philosophical Logic*, 36(2) (2007): 227–247.

[5] Gerhardt, C.I. (ed.), *Die philosophischen Schriften von Gottfried Wilhelm Leibniz*, Volumes i–vii, Berlin: Weidmann, 1875–90.

[6] Henkin, L., 'Completeness in the theory of types', *The Journal of Symbolic*, 15(2) (1950): 81–91.

[7] Kripke, S., 'A completeness theorem in modal logic', *The Journal of Symbolic Logic*, 24(1) (1959): 1–14.

[8] Kripke, S., 'Semantical considerations on modal logic', *Acta Philosophica Fennica*, 16 (1963): 83–94.

[9] Leibniz, G.W., 'A study in the calculus of real addition', in G. Parkinson (ed.), *Leibniz: Logical Papers*, Oxford: Clarendon, 1966, pp. 131–144.

[10] Leibniz, G.W., *Theodicy*, New Haven: Yale University Press, New Haven, 1952.

[11] Lewis, D., 'Counterpart theory and quantified modal logic', *The Journal of Philosophy*, 54(5) (1968): 113–126.

[12] Montague, R., 'The proper treatment of quantification in ordinary English', in: K.J.J. Hintikka, J.M.E. Moravcsik, and P. Suppes (eds.), *Approaches to Natural Language*, D. Reidel, Dordrecht, 1973, pp. 221–242.

[13] Nodelman, U. and E. Zalta, 'Foundations for mathematical structuralism', *Mind*, 123(489) (2014): 39–78.

[14] Oppenheimer, P. and E. Zalta, 'Relations versus functions at the foundations of logic: Type-theoretic considerations', *Journal of Logic and Computation*, 21 (2011): 351–374.

[15] Parkinson, G. (ed.), *Leibniz: Logical Papers*, Oxford: Clarendon, 1966.

[16] Parkinson, G. (ed.), *Leibniz: Philosophical Writings*, London: Dent & Sons, 1973.

[17] Pelletier, F. and E. Zalta, 'How to say goodbye to the third man', *Noûs*, 34(2) (2000): 165–202.

[18] Russell, B., 'On denoting', *Mind*, 14 (1905): 479–493.

[19] Zalta, E., *Abstract Objects: An Introduction to Axiomatic Metaphysics*, Dordrecht: D. Reidel, 1983.

[20] Zalta, E., *Intensional Logic and the Metaphysics of Intentionality*, Cambridge, MA: MIT Press, 1988.

[21] Zalta, E., 'Logical and analytic truths that are not necessary', *The Journal of Philosophy*, 85(2) (1988): 57–74.

[22] Zalta, E., 'Twenty-five basic theorems in situation and world theory', *Journal of Philosophical Logic*, 22(4) (1993): 385–428.

[23] Zalta, E., 'Natural numbers and natural cardinals as abstract objects: A partial reconstruction of Frege's *Grundgesetze* in object theory', *Journal of Philosophical Logic* 28(6) (1999): 619–660.

[24] Zalta, E., 'A (Leibnizian) theory of concepts', *Philosophiegeschichte und logische Analyse/Logical Analysis and History of Philosophy*, 3 (2000): 137–183.

Acknowledgments. The second and third authors would like to thank Branden Fitelson for collaborating on an earlier automated reasoning project and making us aware of the advances in theorem-proving technology.