

STEPS TOWARD A COMPUTATIONAL METAPHYSICS

BRANDEN FITELSON

EDWARD N. ZALTA

Department of Philosophy

CSLI

University of California–Berkeley

Stanford University

branden@fitelson.org

zalta@stanford.edu

http://fitelson.org/

http://mally.stanford.edu/zalta.html

Car si nous l'avions telle que je la conçois, nous pourrions raisonner en metaphysique et en morale à pue pres comme en Geometrie et en Analyse (Leibniz 1890, 21)

If we had it [a *characteristica universalis*], we should be able to reason in metaphysics and morals in much the same way as in geometry and analysis. (Russell 1900, 169)

Quo facto, quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos Computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo . . . dicere: calculemus. (Leibniz 1890, 200)

If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down to their slates, and to say to each other . . . : Let us calculate. (Russell 1900, 170)

The Theory of Abstract Objects I: Language

- Object variables and constants: $x, y, z, \dots; a, b, c, \dots$
- Relation variables and constants: $F^n, G^n, H^n, \dots; P^n, Q^n, R^n, \dots$ (when $n \geq 0$); p, q, r, \dots (when $n=0$)
- Distinguished 1-place relation: $E!$ (read: *concrete*)
- Atomic formulas:
 - $F^n x_1 \dots x_n$ (' x_1, \dots, x_n exemplify F^n ')
 - $x F^1$ (' x encodes F^1 ')
- Complex Formulas: $\neg\varphi, \varphi \rightarrow \psi, \forall\alpha\varphi$ (α any variable), $\Box\varphi$
- Complex Terms:
 - Descriptions: $\iota x\varphi$
 - λ -predicates: $[\lambda x_1 \dots x_n \varphi]$ (φ no encoding subformulas)

The Theory of Abstract Objects: Definitions

- $\&, \vee, \equiv, \exists$, and \diamond are all defined in the usual way
- $O! =_{df} [\lambda x \diamond E!x]$ ('ordinary')
- $A! =_{df} [\lambda x \neg \diamond E!x]$ ('abstract')
- $x =_E y =_{df} O!x \& O!y \& \Box \forall F (Fx \equiv Fy)$
- $x = y =_{df} x =_E y \vee (A!x \& A!y \& \Box \forall F (xF \equiv yF))$
- $F^1 = G^1 =_{df} \Box \forall x (xF^1 \equiv xG^1)$
- $F^n = G^n =_{df}$ (where $n > 1$)
 - $\forall x_1 \dots \forall x_{n-1} ([\lambda y F^n y x_1 \dots x_{n-1}] = [\lambda y G^n y x_1 \dots x_{n-1}] \&$
 - $[\lambda y F^n x_1 y x_2 \dots x_{n-1}] = [\lambda y G^n x_1 y x_2 \dots x_{n-1}] \& \dots \&$
 - $[\lambda y F^n x_1 \dots x_{n-1} y] = [\lambda y G^n x_1 \dots x_{n-1} y])$
- $p = q =_{df} [\lambda y p] = [\lambda y q]$

The Theory of Abstract Objects: Logic

- Simplest second-order quantified S5 modal logic:
1st and 2nd order Barcan formulas (i.e., fixed domains)
free logic for descriptions; modal logic adjusted for rigidity.
- Logic of Encoding: $\diamond xF \rightarrow \Box xF$
- Logic of Identity: $\alpha = \beta \rightarrow [\varphi(\alpha, \alpha) \equiv \varphi(\alpha, \beta)]$
(β substitutable for α)
- Logic of λ -Predicates: (β, η , and α conversion)
 $[\lambda x_1 \dots x_n \varphi]y_1 \dots y_n \equiv \varphi_{x_1, \dots, x_n}^{y_1, \dots, y_n}$ (φ free of descriptions)
 $[\lambda x_1 \dots x_n F^n x_1 \dots x_n] = F^n$
 $[\lambda x_1 \dots x_n \varphi] = [\lambda x'_1 \dots x'_n \varphi']$ (φ, φ' alphabetic variants)
- Logic of Descriptions (contingent logical truth):
 $\psi_z^{ix\varphi} \equiv \exists x(\varphi \ \& \ \forall y(\varphi_x^y \rightarrow y = x) \ \& \ \psi_z^x)$ (ψ atomic/identity)

The Theory Proper

A. Proper Axioms

- $O!x \rightarrow \Box \neg \exists F xF$
- $\exists x(A!x \ \& \ \forall F(xF \equiv \varphi))$, where φ has no free x s

B. Well-Defined Descriptions

- $ix(A!x \ \& \ \forall F(xF \equiv \varphi))$

C. Proper Theorem Schema

- $ix(A!x \ \& \ \forall F(xF \equiv \varphi))G \equiv \varphi_F^G$

Some Examples of Abstract Objects

- *The Complete Concept of y* =
 $ix(A!x \ \& \ \forall F(xF \equiv Fy))$
- *The Actual World* =
 $ix(A!x \ \& \ \forall F(xF \equiv \exists p(p \ \& \ F = [\lambda y p])))$
- *The Truth Value of p* =
 $ix(A!x \ \& \ \forall F(xF \equiv \exists q(q \equiv p \ \& \ F = [\lambda y q])))$
- *The Extension of the Concept G* =
 $ix(A!x \ \& \ \forall F(xF \equiv \forall y(Fy \equiv Gy)))$
- *The Form of G* =
 $ix(A!x \ \& \ \forall F(xF \equiv \Box \forall y(Gy \rightarrow Fy)))$

Example Theorems: World Theory

- $x \vDash p$ ($'p$ is true in x') = $_{df}$ $x[\lambda y p]$
- $World(x)$ = $_{df}$ $\diamond \forall p(x \vDash p \equiv p)$
- $Maximal(x)$ = $_{df}$ $\forall p(x \vDash p \vee x \vDash \neg p)$
- $Consistent(x)$ = $_{df}$ $\neg \exists p[x \vDash (p \ \& \ \neg p)]$
- $Actual(x)$ = $_{df}$ $\forall p(x \vDash p \rightarrow p)$
- **Theorem:** $\forall x(World(x) \rightarrow Maximal(x))$
- **Theorem:** $\forall x(World(x) \rightarrow Consistent(x))$
- **Theorem:** $\exists !x(World(x) \ \& \ Actual(x))$
- **Theorem:** $\Box p \equiv \forall w(w \vDash p)$

Implementation in OTTER I

- Ed’s Theory of Abstract Objects is couched in a second-order language with predicates, modal operators, λ -expressions, and definite descriptions). This poses several challenges for OTTER representation and implementation.
- OTTER is an automated reasoning system which supports full first-order functional and predicate calculus with equality (no second-order, no λ s).
- The main challenge here is one of representation. We must translate claims from Ed’s language into OTTER’s language. Here, we use three key tricks:
 1. Second-Order \mapsto Multi-Sorted First-Order: instead of quantifying over properties, propositions, etc., we quantify over a single domain, and introduce OTTER predicates to sort the domain into properties, objects, etc.
 2. Modal (S5) \mapsto quantified statements over “points” (Ohlbach 1993)
 3. λ s \mapsto complex terms involving functors [(Quine 1960), (Robinson 1970)]

Implementation in OTTER II

- Basic Notation (OTTER syntax in parens):

Predicates	$A, B, C (A, B, C)$	Constants	$a, b, c (a, b, c)$
Variables	$x, y, z (x, y, z)$	Functions	$f, g, h (f, g, h)$
Quantifiers	$\forall, \exists (NA)$	Connectives	$\&, \rightarrow, \vee, \neg, = (NA, NA, , -, =)$

- Formulas vs Clauses (quantifier elimination and CNF)

Formula	Clause (OTTER — Q -free, and CNF)
$(\forall x)(Px \rightarrow Qx)$	$\neg P(x) \mid Q(x) .$
$(\exists x)(Px \& Qx)$	$P(a) . Q(a) .$ (two clauses, new “a”)
$(\forall x)(\exists y)(Rxy \vee x \neq y)$	$R(x, f(x)) \mid \neg(x = f(x)) .$ (new “f”)
$(\forall x)(\forall y)(\exists z)(Rxyz \& Rzyx)$	$R(x, y, f(x, y)) . R(f(x, y), x, y) .$ (new “f”)

- See chapters 1 and 10 of (Kalman 2001), and McCune’s OTTER user manual (McCune 1994) for details on OTTER’s clause notation and syntax.

Implementation in OTTER III

- OTTER implements many rules of inference and strategies (Kalman 2001). For our purposes, it will suffice to discuss just one of these.
- *Hyperresolution* (Kalman 2001, chapter 2) is a generalization of disjunctive syllogism in classical logic. Here are some examples:

$$\begin{array}{ccc} \neg P \mid M. & \neg P(x) \mid M(x). & \neg L(x, f(b)) \mid L(x, f(a)). \\ P. & P(x). & L(y, f(y)). \\ \hline \therefore M. & \therefore M(x). & \therefore L(b, f(a)). \end{array}$$

$$\neg P(x) \mid M(x).$$

- Note: $\frac{P(x).}{\therefore M(a).}$ is *not* a valid hyperresolution inference!

Substitution instances must be *most general*. The Unification Theorem for first-order logic (Robinson 1963) ensures a *unique* most general unifier for each resolution inference. This makes resolution feasible.

Implementation in OTTER IV

- OTTER establishes the validity of first-order arguments *via reductio ad absurdum*: OTTER reasons from the conjunction of the premises and the denial of the conclusion of a valid argument to a contradiction.
- OTTER’s Main Loop (McCune 1994; Wos et al. 1992; Kalman 2001):
 0. Begin with the premises of the (valid) argument in the `usable` list, and the denial of the conclusion of the argument in the `sos` (set of support) list.
 1. Add the denial of the conclusion to the `usable` list.
 2. Using inference rules, *e.g.*, hyperresolution (and/or other forms of resolution), equality rules (and/or other rules), infer all clauses you can.
 3. Process the clauses (check for subsumption, apply restriction strategies, *etc.*), discard unusable ones, and add the remaining ones to the `sos` list.
 4. Pick another member of the `sos` list (using a heuristic – default is “pick shortest” or “best first” – others can be used), and add it to the `usable` list.
 5. Repeat steps 2 – 4 until you reach a contradiction. □

Implementation in OTTER V

Here's a simple OTTER proof of the validity of the following argument:

$$\forall x(\text{Greek}(x) \rightarrow \text{Person}(x)).$$

$$\forall x(\text{Person}(x) \rightarrow \text{Mortal}(x)).$$

$$\text{Greek}(\text{socrates}).$$

$$\text{Mortal}(\text{socrates})$$

1 [] -Greek(x) | Person(x)

2 [] -Person(x) | Mortal(x)

3 [] Greek(socrates)

4 [] -Mortal(socrates)

5 [hyper, 3, 1] Person(socrates)

6 [hyper, 5, 2] Mortal(socrates)

7 [hyper, 6, 4] F

Implementation in OTTER VI

- Ed's second-order theory must be represented in OTTER's first-order language with at least two *sorts*: Property and Object.
- E.g., one-place exemplification Fx and encoding xF (Ed's two forms of predication) can be represented and typed in OTTER as follows:
 - all $F x$ ($\text{Ex1}(F, x) \rightarrow \text{Property}(F) \ \& \ \text{Object}(x)$).
 - all $F x$ ($\text{Enc}(x, F) \rightarrow \text{Property}(F) \ \& \ \text{Object}(x)$).
- Two-place predication requires a new relation: $\text{Ex2}(R, x, y)$, etc.
- Modal (S5) claims can be translated into OTTER Kripke-style (Ohlbach 1993), with the use of a third sort: Point (*not World!*).
 - all $F x w$ ($\text{Ex1}(F, x, w) \rightarrow \text{Property}(F) \ \& \ \text{Object}(x) \ \& \ \text{Point}(w)$).

Implementation in OTTER VII

- Propositions can't be defined as 0-place properties (OTTER has no such), so a fourth sort of term is required: **Proposition**.
- With sorted terms, OTTER requires explicit typing conditions:
 - all x ($\text{Property}(x) \rightarrow \text{-Object}(x)$).
 - all x ($\text{Property}(x) \rightarrow \text{-Proposition}(x)$).
 - all x ($\text{Property}(x) \rightarrow \text{-Point}(x)$).
- Complex properties (i.e., λ -expressions) can be represented in OTTER using functors [(Quine 1960), (Robinson 1970)]. E.g., we represent the property *being such that p* ($[\lambda y p]$) using a functor VAC:
 - all p ($\text{Proposition}(p) \leftrightarrow \text{Property}(\text{VAC}(p))$).
 - all $x p w$ ($(\text{Object}(x) \ \& \ \text{Proposition}(p) \ \& \ \text{Point}(w)) \rightarrow \text{Ex1}(\text{VAC}(p), x, w) \leftrightarrow \text{True}(p, w)$).

Why OTTER?

- Question #1: Why did we choose a first-order system like OTTER, instead of a second-order system like NQTHM (Boyer and Moore 1979) or ISABELLE/HOL (Nipkow et al. 2002)?
- The main problem here is that there is no unification theorem (Robinson 1963) for second order logic. This makes searching for proofs of any depth almost hopeless in higher order systems.
- For this reason, higher-order systems have been used almost exclusively for proof *checking* or *verification*, but not for *finding* proofs of non-trivial depth. See (Kunen 1998) for discussion.
- We (Leibniz) wanted a mechanical way of *discovering* proofs in Metaphysics — not merely *checking* proofs we already knew.
- Question #2: Why OTTER in particular? OTTER is the most flexible, powerful, and robust first-order system that is freely available.

Successes & Challenges

- We've used OTTER to find proofs (without guiding OTTER's search using known lemmas) of all but one of the theorems reported in Ed's paper (Zalta 1993) on World Theory and Situation Theory.
 - The only theorem in this context not yet proven with OTTER is the \Leftarrow direction of $\lceil \Box p \equiv \forall w(w \vDash p) \rceil$ — the last theorem on slide 7.
- We've used OTTER to find proofs (again, without guiding OTTER's search) of all of the theorems reported Ed's (and Jeff Pelletier's) paper (Pelletier and Zalta 2000) on the Third Man Argument.
- Current Challenge: Using OTTER to find proofs of theorems reported by Ed in (Zalta 2000), concerning Leibniz's theory of concepts.
- Next Challenge: Use OTTER to find proofs of theorems reported by Ed in (Zalta 1999), concerning Frege's theory of \mathbb{N} in the *Grundgesetze*.

An OTTER Proof That Every World Is Maximal

```

1 [] -Proposition(x) | Proposition(~x)
2 [] -Point(x) | -Proposition(y) | True(~y,x) | True(y,x)
3 [] -Object(x) | -Situation(x) | Maximal(x) | Proposition(f1(x))
4 [] -Object(x) | -Situation(x) | Maximal(x) | -TrueIn(f1(x),x)
5 [] -Object(x) | -Situation(x) | Maximal(x) | -TrueIn(~f1(x),x)
6 [] -Object(x) | -World(x) | Situation(x)
7 [] -Object(x) | -World(x) | Point(f2(x))
8 [] -Object(x) | -World(x) | -Proposition(y) | TrueIn(y,x) | -True(y,f2(x))
9 [] -World(x) | Object(x)
10 [] World(c1)
11 [] -Maximal(c1)
12 [hyper,9,10] Object(c1)
13 [hyper,6,12,10] Situation(c1)
14 [hyper,7,12,10] Point(f2(c1))

```

```

15 [hyper,3,12,13] Maximal(c1) | Proposition(f1(c1))
16 [hyper,15,1] Maximal(c1) | Proposition(~f1(c1))
27 [hyper,2,14,15] True(~f1(c1),f2(c1)) | True(f1(c1),f2(c1)) | Maximal(c1)
32 [hyper,27,11] True(~f1(c1),f2(c1)) | True(f1(c1),f2(c1))
33 [hyper,32,8,12,10,16] True(f1(c1),f2(c1)) | TrueIn(~f1(c1),c1) | Maximal(c1)
35 [hyper,33,11] True(f1(c1),f2(c1)) | TrueIn(~f1(c1),c1)
36 [hyper,35,5,12,13] True(f1(c1),f2(c1)) | Maximal(c1)
37 [hyper,36,11] True(f1(c1),f2(c1))
38 [hyper,37,8,12,10,15] TrueIn(f1(c1),c1) | Maximal(c1)
39 [hyper,38,11] TrueIn(f1(c1),c1)
40 [hyper,39,4,12,13] Maximal(c1)
42 [hyper,40,11] F

```

References

- Boyer, R. and J. Moore (1979). *A computational logic*. New York: Academic Press [Harcourt Brace Jovanovich]. ACM Monograph Series.
- Kalman, J. (2001). *Automated Reasoning with Otter*. Princeton, N.J.: Rinton Press.
- Kunen, K. (1998). Nonconstructive computational mathematics. *Journal of Automated Reasoning* 21(1), 69–97.
- Leibniz, G. (1890). In C. Gerhardt (Ed.), *Die philosophischen Schriften von Gottfried Wilhelm Leibniz*, Volume vii. Berlin: Olms.
- McCune, W. (1994). Otter 3.0 Reference Manual and Guide. Tech. Report ANL-94/6, Argonne National Laboratory, Argonne, IL. Also see <http://www.mcs.anl.gov/AR/otter/>.
- Nipkow, T., L. Paulson, and M. Wenzel (2002). *Isabelle/HOL: A proof assistant for higher-order logic*. Lecture Notes in Computer Science (v.2283). Berlin: Springer.
- Ohlbach, H. (1993). Translation methods for non-classical logics: an overview. *Bulletin of the Interest Group in Pure and Applied Logics* 1(1), 69–89.
- Pelletier, F. and E. Zalta (2000). How to say goodbye to the third man. *Nous* 34(2), 165–202.

Quine, W. (1960). Variables explained away. In *Selected Logical Papers*, pp. 227–235. New York: Random House.

Robinson, J. A. (1963). Theorem-proving on the computer. *Journal of the Association of Computing Machinery* 10, 163–174.

Robinson, J. A. (1970). A note on mechanizing higher order logic. In *Machine Intelligence*, 5, pp. 123–133. American Elsevier, New York.

Russell, B. (1900). *A Critical Exposition of the Philosophy of Leibniz*. Cambridge: CUP.

Wos, L., R. Overbeek, E. Lusk, and J. Boyle (1992). *Automated Reasoning: Introduction and Applications*, 2nd edition. New York: McGraw-Hill.

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

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

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