# Towards Leibniz's Goal of a Computational Metaphysics

Edward N. Zalta

Center for the Study of Language and Information

Stanford University

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

# Motive and Methods

Leibniz:

*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.*

*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.*

*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**.

# The Value of Computational Metaphysics

- Discover countermodels to hypotheses and detect errors in reasoning.
- Discover facts about the strength of axioms and premises needed to derive metaphysical conclusions
- Confirm premise consistency and find smallest models of metaphysical claims.
- Derive interesting theorems and confirm valid reasoning.
- Clarify epistemological issues in light of the metaphysical and logical results.
- Methodology:
  - Download/install automated reasoning engine, e.g., Prover9: http://www.cs.unm.edu/~mccune/mace4/
  - Test it/develop an understanding of how it works
  - Represent logical and non-logical premises.
  - Find/investigate proofs, premise sets, and models/countermodels

# Logic → Prover9 Syntax → Clausal Normal Form

- $\forall x(Px \rightarrow Qx)$ (Logic)

  `all x (Px -> Qx).` (Prover9 syntax)

  `-P(x) | Q(x).` (Clausal Normal Form)

- $\exists x(Px \,\&\, Qx)$

  `exists x (Px & Qx).`

  `P(c1).`

  `Q(c1).`

- $\forall x \exists y(Rxy \vee x \neq y)$

  `all x exists y (Rxy | -(x=y)).`

  `R(x,f(x)) | -(x = f(x)).`

- $\forall x \forall y \exists z(Rxyz \,\&\, Rzyx)$

  `all x all y exists z (Rxyz & Rzyx)`

  `R(x,y,f(x,y)).`

  `R(f(x,y),x,y).`

# PROVER9

- PROVER9 establishes the validity of first-order arguments *via reductio ad absurdum*. Its main loop:
  1. Let the premises of the argument constitute the *usable* list.
  2. Add the negation of the conclusion to the *set of support* list.
  3. While the sos list is not empty:
     1. Select a given clause from sos and move it to the usable list;
     2. Infer new clauses using the inference rules in effect; each new clause must have the given clause as one of its parents and members of the usable list as its other parents;
     3. process each new clause;
     4. append new clauses that pass the retention tests to the sos list.
  4. End of while loop.
  5. Cycle until you reach a contradiction.
- See http://www.cs.unm.edu/~mccune/prover9/manual/2008-11A/

# Example

- Argument:

$$\frac{\begin{array}{l}\forall x(Greek(x) \rightarrow Person(x)) \\ \forall x(Person(x) \rightarrow Mortal(x)) \\ Greek(s)\end{array}}{Mortal(s)}$$

- Input file:

```
all x (Greek(x) -> Person(x)).
all x (Person(x) -> Mortal(x)).
Greek(s).
Mortal(s).
```

- PROVER9 proof:

```
1 [ ] -Greek(x) | Person(x)
2 [ ] -Person(x) | Mortal(x)
3 [ ] Greek(s)
4 [ ] -Mortal(s)
5 [hyper,3,1] Person(s)
6 [hyper,5,2] Mortal(s)
7 [hyper,6,4] F
```

# Leibniz's Calculus of Primitive Concepts

- Leibniz 1690: axioms for $x \oplus y$ and $x \leq y$.
- Proposition 12: $\forall x, y, z(y \leq z \rightarrow x \oplus y \leq x \oplus z)$

```
all x all y all z (IsIn(y,z) -> IsIn(Sum(x,y),Sum(x,z))).
```

- Premises:

```
all x all y all z (Sum(Sum(x,y),z) = Sum(x,Sum(y,z))).
```

Associativity

```
all x all y (IsIn(x,y) <-> (exists z (Sum(x,z) = y))).
```

Definition

- Demo in PROVER9.
  - Note premise 1 is clausified in line 4 and premise 2 is clausified in lines 5 and 6.
  - Note the Skolem function in clause for premise 2.
  - Note all the premises are used in the proof.
- Note that the proof fails without Associativity. Demo countermodel using MACE4: it postulates 3 entities (`c1`, `c2`, and `c3`), for the negation of the conclusion.
- Mistake: Leibniz omitted Associativity from his list of axioms.

# Countermodel to 'Theorem' About Plato's Forms

- In Pelletier and Zalta 2000:
  - *The Form of F* ('$\Phi_F$') $=_{df}$ $\iota x(A!x \mathbin{\&} \forall G(xG \equiv F \Rightarrow G))$
  - $Participates_{\mathrm{PTA}}(x, y) \ =_{df} \ \exists F(y = \Phi_F \mathbin{\&} Fx)$
  - $Participates_{\mathrm{PH}}(x, y) \ =_{df} \ \exists F(y = \Phi_F \mathbin{\&} xF)$
- We alleged the following was a theorem:
  - $xF \equiv Participates_{\mathrm{PH}}(x, \Phi_F)$
- Mace found an countermodel to the right-to-left direction.
  - Choose $P$ = being-$Q$-and-not-$Q$ (for any $Q$)
    Choose $T$ = being-round-and-square.
  - In object theory, one can consistently assert $P \neq T$.
  - Even when $P \neq T$, it is provable in object theory that $\Phi_P = \Phi_T$
  - Let $b$ be the abstract object that encodes exactly one property, namely, $T$.
- It follows that:
  - $Participates_{\mathrm{PH}}(b,\Phi_P)$ since $Participates_{\mathrm{PH}}(b, \Phi_T)$ and $\Phi_P = \Phi_T$.
  - $\neg bP$.

# Ontological Argument Premise Set

in Oppenheimer and Zalta 1991

- One Logical Axiom:
  - Russell (1905) Axiom
    $\psi_z^{\imath x\varphi} \equiv \exists y(\varphi_x^y \ \&\ \forall u(\varphi_x^u \rightarrow u\!=\!y) \ \&\ \psi_z^y)$
- Three Non-Logical Axioms:
  - *Connectedness:* $\forall x, y(Gxy \ \vee\ Gyx \ \vee\ x\!=\!y)$
  - *Premise 1:* $\exists x(Cx \ \&\ \neg\exists y(Gyx \ \&\ Cy)) \quad (= \exists x\varphi_1)$
  - *Premise 2:* $\neg E!\imath x\varphi_1 \rightarrow \exists y(Gy\imath x\varphi_1 \ \&\ Cy)$
- Definition:
  - God (‘$g$’) $=_{df}\ \imath x\varphi_1$

## Logical Theorems: Consequences of the Logical Axiom

- *Description Theorem 1:* $\exists!x\varphi \rightarrow \exists y(y = \imath x\varphi)$
- *Lemma 1:* $\tau = \imath x\varphi \rightarrow \varphi^{\tau}_{x}$, for any term $\tau$
- *Description Theorem 2:* $\exists y(y = \imath x\varphi) \rightarrow \varphi^{\imath x\varphi}_{x}$
- *Lemma 2:* $\exists x\varphi_1 \rightarrow \exists!x\varphi_1$

# Proof of Lemma 2

| | | |
|---|---|---|
| 1 | $\exists x(Cx \mathbin{\&} \neg\exists y(Gyx \mathbin{\&} Cy))$ | Assume antecedent. |
| 2 | $Ca \mathbin{\&} \neg\exists y(Gya \mathbin{\&} Cy)$ | '$a$' arbitrary |
| 3 | $\exists z(z \neq a \mathbin{\&} Cz \mathbin{\&} \neg\exists y(Gyz \mathbin{\&} Cy))$ | Assume for reductio. |
| 4 | $b \neq a \mathbin{\&} Cb \mathbin{\&} \neg\exists y(Gyb \mathbin{\&} Cy)$ | '$b$' arbitrary |
| 5 | $Gab \lor Gba \lor a = b$ | by Meaning Postulate |
| 6 | $Gab \lor Gba$ | from 5, and 4 ($a \neq b$) |
| 7 | $Gab$ | Assumption |
| 8 | $Gab \mathbin{\&} Ca$ | given $Ca$ in 2 |
| 9 | $\exists y(Gyb \mathbin{\&} Cy)$ | Contradicts 4 |
| 10 | $Gba$ | Assumption |
| 11 | $Gba \mathbin{\&} Cb$ | given $Cb$ in 4 |
| 12 | $\exists y(Gya \mathbin{\&} Cy)$ | Contradicts 2 |
| 13 | $\neg\exists z(z \neq a \mathbin{\&} Cz \mathbin{\&} \neg\exists y(Gyz \mathbin{\&} Cy))$ | By reductio |

# The Ontological Argument (1991)

1. $\exists x\varphi_1$ — Premise 1
2. $\exists!x\varphi_1$ — from (1), by Lemma 2: $\exists x\varphi_1 \rightarrow \exists!x\varphi_1$
3. $\exists y(y = \iota x\varphi_1)$ — from (2), by Description Thm 1: $\exists!x\varphi \rightarrow \exists y(y = \iota x\varphi)$
4. $C\iota x\varphi_1 \ \& \ \neg\exists y(Gy\iota x\varphi_1 \ \& \ Cy)$ — from (3), by Description Thm 2: $\exists y(y = \iota x\varphi) \rightarrow \varphi_x^{\iota x\varphi}$
5. $\neg E!\iota x\varphi_1$ — Assumption, for *reductio*
6. $\exists y(Gy\iota x\varphi_1 \ \& \ Cy)$ — from (5), by Premise 2: $\neg E!\iota x\varphi_1 \rightarrow \exists y(Gy\iota x\varphi_1 \ \& \ Cy)$
7. $\neg\exists y(Gy\iota x\varphi_1 \ \& \ Cy)$ — from line (4), by &E
8. $E!\iota x\varphi_1$ — from lines (5)–(7), by *reductio*
9. $E!g$ — from line (8), by df of $g$

# Implementation in PROVER9

- The ontological argument relies on instances of logical axiom schemata (e.g., for descriptions). But PROVER9 is a first-order automated reasoning system that can't process schemata.
- Strategy: Treat schemata as second-order statements and then translate the relevant ones into two-sorted first-order logic.
- Example: 2nd-order statement → 2-sorted 1st-order logic.
- $\forall F \forall x \forall y(Fx \equiv Fy)$ translates to

```
all F all x all y ((Property(F) & Object(x) & Object(y)) ->
  (Ex1(F,x) <-> Ex1(F,y)))
```

- Use sorting on one-place and two-place predications $Fx$ and $Rxy$:

```
all F all x (Ex1(F,x) -> Property(F) & Object(x)).
all F all x (Ex2(R,x,y) -> Relation(R) & Object(x) & Object(y)).
```

- We thus quantify over a single domain, and introduce PROVER9 predicates to sort the domain into properties, objects, etc.

## A Problem With Descriptions

- A subtlety: It would seem natural to represent formulas with definite descriptions such as $G\imath xFx$ as:

  `x=The(F) & Gx`
- But we have represented them as:

  `Is_The(x,F) & Gx`
- The reason we don't use the former is that an inconsistency would otherwise arise between the following two principles for sorting on `Object`, `Property`, and `The(F)`:

  `all x (Object(x) -> -Property(x))`

  `all x all F (x=The(F) -> (Object(x) & Property(F)))`
- For when `Object(b)`, the first implies `-Property(b)`, and the second implies: `all x -(x=The(b))`. But instantiating to `The(b)` yields a contradiction with the law of identity `-(The(b)=The(b))`.

# Premise 1

- Definition of `None_greater`:

```
all x (Object(x) -> (Ex1(none_greater,x) <->
  (Ex1(conceivable,x) & -(exists y (Object(y) &
  Ex2(greater_than,y,x) & Ex1(conceivable,y)))))).
```

- This clausifies to:

```
-Object(x) | -Ex1(none_greater,x) | Ex1(conceivable,x).
-Object(x) | -Ex1(none_greater,x) | -Object(y) | -Ex2(greater_than,y,x) | -Ex1(conceivable,y).
-Object(x) | Ex1(none_greater,x) | -Ex1(conceivable,x) | Object(f3(x)).
-Object(x) | Ex1(none_greater,x) | -Ex1(conceivable,x) | Ex2(greater_than,f3(x),x).
-Object(x) | Ex1(none_greater,x) | -Ex1(conceivable,x) | Ex1(conceivable,f3(x)).
```

- Premise 1: $\exists x(Cx \mathbin{\&} \neg\exists y(Gyx \mathbin{\&} Cy))$ becomes

```
exists x (Object(x) & Ex1(none_greater,x)).
```

- See chapters 1, 10 of Kalman, and McCune's Prover9 user manual for details on Prover9's clause notation and syntax.

# Ontological Argument Input File for Prover9

- The ontological argument input file:

http://mally.stanford.edu/cm/ontological-argument/ontological.in

- Save the file and run it:

  Graphically: cut and paste.

  Command line: `prover9 < ontological.in`

- Command line run shows the clauses!

http://mally.stanford.edu/cm/ontological-argument/ontological-clauses.txt

- Find out which clauses are used in the proof and identify the source premise.

# Prover9's Ontological Argument

- What did it use?

  Description Theorem 1: No!

  Description Theorem 2: It used the one clause.

  Sorting on `Is_the`: It used one of two clauses.

  Definition `none_greater`: It used one of five clauses.

  Premise 1: No!

  Lemma 2: No!

  Premise 2: It used all three clauses.

  Definition of God: It used the one clause.
- It has found a simpler proof!
- Task: reverse engineer the proof into something a human would find readable.

# Ontological Argument Reduced Premise Set

(Reverse Engineered from Prover9)

- The Russell (Logical) Axiom for Descriptions, which yields:
  - Description Theorem 2: $\exists y(y = \iota x\varphi) \rightarrow \varphi_x^{\iota x\varphi}$
  - Description Theorem 3: $\psi_x^{\iota x\varphi} \rightarrow \exists y(y = \iota x\varphi)$,
    where $\psi$ is any atomic formula with $x$ free.
- (Nonlogical) Premise 2: $\neg E!\iota x\varphi_1 \rightarrow \exists y(Gy\iota x\varphi_1 \ \& \ Cy)$
- Definition: $g \ =_{df} \ \iota x\varphi_1$

# Deus Ex Machina

| | | |
|---|---|---|
| 1. | $\neg E!\iota x\varphi_1$ | Assumption for *reductio* |
| 2. | $\exists y(Gy\iota x\varphi_1\ \&\ Cy)$ | from (1), by MP and Premise 2: $\neg E!\iota x\varphi_1 \rightarrow \exists y(Gy\iota x\varphi_1\ \&\ Cy)$ |
| 3. | $Gb\iota x\varphi_1\ \&\ Cb$ | from (2), by EE, ‘$b$’ arbitrary |
| 4. | $Gb\iota x\varphi_1$ | from (3), by &E |
| 5. | $\exists y(y = \iota x\varphi_1)$ | from (4), by Description Theorem 3: $\psi_x^{\iota x\varphi} \rightarrow \exists y(y = \iota x\varphi)$ |
| 6. | $C\iota x\varphi_1\ \&\ \neg\exists y(Gy\iota x\varphi_1\ \&\ Cy)$ | from (5), by Description Theorem 2: $\exists y(y = \iota x\varphi) \rightarrow \varphi_x^{\iota x\varphi}$ |
| 7. | $\neg\exists y(Gy\iota x\varphi_1\ \&\ Cy)$ | from (6), by &E |
| 8. | Contradiction | from (2), (7) by &I |
| 9. | $E!\iota x\varphi_1$ | from (1)–(8), by *reductio* |
| 10. | $\exists y(y = \iota x\varphi_1)$ | from (9), by Description Theorem 3: $\psi_x^{\iota x\varphi} \rightarrow \exists y(y = \iota x\varphi)$ |
| 11. | $E!g$ | from (9), (10), by definition ‘$g$’, UE and =E, in free logic. |

The argument rests solely on Premise 2: $\neg E!\iota x\varphi_1 \rightarrow \exists y(Gy\iota x\varphi_1\ \&\ Cy)$

## Check For Models of the Premises

- Input file:

http://mally.stanford.edu/cm/ontological-argument/ontological-model.in

- Run it graphically or via command line:

  Graphically: cut out conclusion, run Mace, show Cooked view.

  Command line: `mace -c -N 8 -p 1 < ontological-model.in`

- Simplest model equates the existence predicate '`e`' with a model element 0, and equates '`g`' (God) with that same element. Prover9 doesn't know objects aren't relations, that `e` is a `Relation1`, or that if `x` exemplifies `y`, then `x` is an `Object` and `y` is a `Relation1`.

```
all x (Object(x) -> -Relation1(x)).
Relation1(e).
all x all F (Ex1(F,x) -> (Relation1(F) & Object(x))).
```

- Check the model. Continue the process to get an *intended* model!

## Mace4 Doesn't Find Intended Model: Use Paradox

- Demo in Prover9
- Use ontological-intended-model.p. Call with paradoxm.

# Object Theory in Prover9: I

- We translate modal claims into quantifications over 'propositions' and 'points'. $\Box p$ becomes:

```
all d (Point(d) -> True(p,d)).
all p all d (True(p,d) -> (Proposition(p) & Point(d))).
```

- Predication requires sorts and is relativized to points:

```
all F all x all d (Ex1At(F,x,d) -> Property(F) & Object(x) & Point(d)).
all F all x all d (EncAt(x,F,d) -> Property(F) & Object(x) & Point(d)).
```

- Rigidity of encoding:

```
all x all F ((Object(x) & Property(F)) ->
  ( (exists d (Point(d) & EncAt(x,F,d))) ->
    (all d (Point(d) -> EncAt(x,F,d))) )).
```

- $\lambda$-expressions $\rightarrow$ functors: *being such that p* becomes `VAC`:

```
all p (Proposition(p) <-> Property(VAC(p))).
all x p w ((Object(x) & Proposition(p) & Point(d)) ->
  (Ex1At(VAC(p),x,d) <-> True(p,d))).
```

# Object Theory in Prover9: II

- Sorting on EncpAt and TrueInAt.

```
all x all p all d (Object(x) & Proposition(p) & Point(d) ->
  (EncpAt(x,p,d) <->
    (exists F (Property(F) & F=VAC(p) & EncAt(x,F,d))))).
```

```
all p all x all d (Object(x) & Proposition(p) & Point(d) ->
  (TrueInAt(p,x,d) <-> EncpAt(x,p,d))).
```

- Prover9 then clasifies everything, e.g., the definition of a world:
  - $PossibleWorld(x) \ =_{df} \ Situation(x)\ \&\ \Diamond\forall p(s \models p \ \equiv \ p)$
- This gets input into prover9 as:
  - ```
    all x all d (Object(x) & Point(d) -> (WorldAt(x,d) <->
     SituationAt(x,d) & (exists d2 (Point(d2) &
     (all p (Proposition(p) -> (TrueInAt(p,x,d) <-> True(p,d2))))))))
    ```
- Prover9 clausifies this to:

```
-Object(x) | -Point(y) | -WorldAt(x,y) | SituationAt(x,y).
-Object(x) | -Point(y) | -WorldAt(x,y) | Point(f1(x,y)).
-Object(x) | -Point(y) | -WorldAt(x,y) | -Proposition(z) | -TrueInAt(z,x,y) | True(z,f1(x,y)).
-Object(x) | -Point(y) | -WorldAt(x,y) | -Proposition(z) | TrueInAt(z,x,y) | -True(z,f1(x,y)).
-Object(x) | -Point(y) | WorldAt(x,y) | -SituationAt(x,y) | -Point(z) | Proposition(f2(x,y,z)).
-Object(x) | -Point(y) | WorldAt(x,y) | -SituationAt(x,y) | -Point(z) | TrueInAt(f2(x,y,z),x,y) | True(f2(x,y,z),z).
-Object(x) | -Point(y) | WorldAt(x,y) | -SituationAt(x,y) | -Point(z) | -TrueInAt(f2(x,y,z),x,y) | -True(f2(x,y,z),z).
```
```

# A PROVER9 Proof That Every World Is Maximal

- Definition of Maximal:
  - $Maximal(x) =_{df} \forall p(x \models p \;\vee\; x \models \neg p)$
  - `all x (Object(x) -> (Maximal(x) <-> Situation(x) & (all p (Proposition(p) -> TrueIn(p,x) | TrueIn(NEG(p),x))))).`
  - Clausifies to:

```
-Object(x) | -Maximal(x) | Situation(x).
-Object(x) | -Maximal(x) | -Proposition(y) | TrueIn(y,x) | TrueIn(NEG(y),x).
-Object(x) | Maximal(x) | -Situation(x) | Proposition(f1(x)).
-Object(x) | Maximal(x) | -Situation(x) | -TrueIn(f1(x),x).
-Object(x) | Maximal(x) | -Situation(x) | -TrueIn(NEG(f1(x)),x).
```

- The claim to be proved is:
  - $\forall x(World(x) \rightarrow Maximal(x))$
  - `all x (World(x) -> Maximal(x)).`
  - `-World(x) | Maximal(x).`

# An PROVER9 Proof That Every World Is Maximal

```
1 (all p (Proposition(p) -> Proposition(NEG(p)))) [assumption].
2 (all d all p (Point(d) & Proposition(p) -> (True(NEG(p),d) <-> -True(p,d)))) [assumption].
3 (all x (Object(x) -> (Maximal(x) <-> Situation(x) & (all p (Proposition(p) -> TrueIn(p,x) | TrueIn(NEG(p),x)))))) [assumption].
4 (all x (Object(x) -> (World(x) <-> Situation(x) & (exists y (Point(y) & (all p (Proposition(p) -> (TrueIn(p,x) <-> True(p,y))))))))) [assumption].
5 (all x (World(x) -> Object(x))) [assumption].
6 (all x (World(x) -> Maximal(x))) [goal].
7 -Object(x) | -World(x) | Point(f2(x)).  [clausify(4)].
9 -Point(x) | -Proposition(y) | True(NEG(y),x) | True(y,x).  [clausify(2)].
13 -Object(x) | Maximal(x) | -Situation(x) | Proposition(f1(x)).  [clausify(3)].
16 -Object(x) | Maximal(x) | -Situation(x) | -TrueIn(f1(x),x).  [clausify(3)].
17 -Object(x) | Maximal(x) | -Situation(x) | -TrueIn(NEG(f1(x)),x).  [clausify(3)].
18 -Maximal(c1).  [deny(6)].
20 -Object(x) | -World(x) | Situation(x).  [clausify(4)].
26 -Object(c1) | -Situation(c1) | Proposition(f1(c1)).  [resolve(18,a,13,b)].
27 -Object(c1) | -Situation(c1) | -TrueIn(f1(c1),c1).  [resolve(18,a,16,b)].
28 -Object(c1) | -Situation(c1) | -TrueIn(NEG(f1(c1)),c1).  [resolve(18,a,17,b)].
29 World(c1).  [deny(6)].
31 -Object(x) | -World(x) | -Proposition(y) | TrueIn(y,x) | -True(y,f2(x)).  [clausify(4)].
32 -World(x) | Object(x).  [clausify(5)].
34 -Object(x) | -World(x) | -Proposition(y) | True(NEG(y),f2(x)) | True(y,f2(x)).  [resolve(7,c,9,a)].
38 -Object(c1) | Proposition(f1(c1)) | -Object(c1) | -World(c1).  [resolve(26,b,20,c)].
39 -Object(c1) | -TrueIn(f1(c1),c1) | -Object(c1) | -World(c1).  [resolve(27,b,20,c)].
40 -Object(c1) | -TrueIn(NEG(f1(c1)),c1) | -Object(c1) | -World(c1).  [resolve(28,b,20,c)].
41 -Proposition(x) | Proposition(NEG(x)).  [clausify(1)].
43 -Object(c1) | -Proposition(x) | TrueIn(x,c1) | -True(x,f2(c1)).  [resolve(29,a,31,b)].
44 Object(c1).  [resolve(29,a,32,a)].
47 -Object(c1) | -Proposition(x) | True(NEG(x),f2(c1)) | True(x,f2(c1)).  [resolve(34,b,29,a)].
48 -Proposition(x) | True(NEG(x),f2(c1)) | True(x,f2(c1)).  [copy(47),unit_del(a,44)].
55 -Object(c1) | Proposition(f1(c1)) | -Object(c1).  [resolve(38,d,29,a)].
56 Proposition(f1(c1)).  [copy(55),merge(c),unit_del(a,44)].
57 -Object(c1) | -TrueIn(f1(c1),c1) | -Object(c1).  [resolve(39,d,29,a)].
58 -TrueIn(f1(c1),c1).  [copy(57),merge(c),unit_del(a,44)].
59 -Object(c1) | -TrueIn(NEG(f1(c1)),c1) | -Object(c1).  [resolve(40,d,29,a)].
60 -TrueIn(NEG(f1(c1)),c1).  [copy(59),merge(c),unit_del(a,44)].
61 -Proposition(x) | TrueIn(x,c1) | -True(x,f2(c1)).  [back_unit_del(43),unit_del(a,44)].
63 True(NEG(f1(c1)),f2(c1)) | True(f1(c1),f2(c1)).  [resolve(56,a,48,a)].
64 Proposition(NEG(f1(c1))).  [resolve(56,a,41,a)].
65 -True(f1(c1),f2(c1)).  [ur(61,a,56,a,b,58,a)].
66 True(NEG(f1(c1)),f2(c1)).  [back_unit_del(63),unit_del(b,65)].
68 F.  [ur(61,a,64,a,b,60,a),unit_del(a,66)].
```

# The Lewis Principle

- $\Diamond p \rightarrow \exists w(w \models p)$
- This gets input into Prover9 as:
  - `Point(W).`
  - `all p (PossiblyTrue(p) <-> (exists d (Point(d) & True(p,d)))).`
  - `all x (World(x) <-> WorldAt(x,W)).`
  - `all x all p (TrueIn(p,x) <-> TrueInAt(p,x,W)).`
  - `all p (Proposition(p) -> (PossiblyTrue(p) -> (exists x (World(x) & TrueIn(p,x))))).`

# A Prover9 Proof of the Lewis Principle

- Prove some preliminary lemmas: http://mally.stanford.edu/cm/worlds/new/
  - `WorldAt(x,d)` is rigid.
  - `TrueInAt(p,x,d)` is rigid.
  - If `WorldAt(x,d) & ActualAt(x,d)`, then for any `p`,

    `TrueInAt(p,x,d) <-> True(p,d)`
- Input file Theorem25a (→): theorem25a.in
- Clausification of the Argument: clauses25a.txt
- (Demo) What did it use?

# Analysis of the Prover9 Proof of the Lewis Principle

- Logical Axioms [Assumption, clauses from 1]
  - `W` is a (distinguished) `Point`
  - (1) Sorting on `WorldAt(x,d)`
- Logical Theorems [clauses from 2, 3, 4]:
  - (2) `WorldAt(x,d)` is rigid.
  - (3) `TrueInAt(p,x,d)` is rigid.
  - (4) If `WorldAt(x,d)` and `ActualAt(x,d)`, then `TrueInAt(p,x,d) <-> True(p,d).`
- Definitions [clauses from 6, 7, 8]:
  - (6) `PossiblyTrue(p)` = `True(p,d)` for some `d`
  - (7) `World(x)` = `WorldAt(x,W)`
  - (8) `TrueIn(p,x)` = `TrueInAt(p,x,W)`
- ‘Proper’ Theorem [clauses from 5]
  - (5) At every point d, there is an actual world at d.

## Theorems in Situation and World Theory

- Computational Metaphysics page on worlds: http://mally.stanford.edu/cm/worlds/new/
- See Fitelson and Zalta 2007.

## Small Models Clarifies Epistemological Questions

- The Lewis Principle ("For every way a world might be there is a world which is that way.") can be proved from object-theoretic premises that are true in a domain of size 2.
- The ontology grows only when you add ordinary modal beliefs: if those are given, the existence of the possible worlds depends only on the axioms and definitions used in the proof.
- To the extent that the Lewis Principle is the core of Lewis' theory, one can argue that his *theoretical views* imply only a small ontology.

# Analysis of Deus Ex Machina

- Computational techniques show Anselm needed only one non-logical premise.
- The premise's antecedent doesn't assume the description denotes.
- The simpler argument justifies the use of the description indirectly.
- The simpler argument is clearly a *diagonal* argument.
- The simpler argument yields an insight about how little content the *greater_than* relation must have for the argument.
- The model-building program MACE4 proved useful; it showed our premise set was consistent.

# Soundness: I

- The soundness of ontological argument turns on the truth of a single premise:
  Premise 2: $\neg E!\iota x\varphi_1 \rightarrow \exists y(G y \iota x\varphi_1 \ \& \ Cy)$
- It appears to be plausible prima facie, and work by Parsons (1980) and others shows one may consistently and coherently take existence to be a predicate, with an extension that is a subset of the domain.
- To show Premise 2 is false, one must argue that the antecedent, $\neg E!\iota x\varphi_1$, is true and that the consequent, $\exists y(G y \iota x\varphi_1 \ \& \ Cy)$, is false.
- There are two different ways for the antecedent of Premise 2 to be true: on the one hand, the description $\iota x\varphi_1$ could fail to denote, in which case, the atomic formula $E!\iota x\varphi_1$ is false and its negation (the antecedent) true; on the other, the description does denote, and the object it denotes fails to have the property of existence.

# Soundness II

1. Suppose $\iota x\varphi_1$ fails to denote and the antecedent of Premise 2 is therefore true. If so, then the consequent is false, on the grounds that if the description fails to denote, then a claim of the form $Gy\iota x\varphi_1$ is false for every $y$ (since it is an atomic formula with a non-denoting term). If $Gy\iota x\varphi_1$ is false for every $y$, then $Gy\iota x\varphi_1 \,\&\, Cy$ is false for every $y$. Therefore, the consequent of Premise 2 is false.
2. Suppose $\iota x\varphi_1$ denotes and the antecedent of Premise 2 is true because the object denoted lacks existence. If the description denotes, then there is a unique thing such that nothing greater can be conceived. So the consequent of Premise 2, which says there is something greater than it, is false.

# Soundness III

- Final case: the description $\iota x\varphi_1$ denotes and the object it denotes exists. Then, the antecedent of Premise 2 is false, making Premise 2 true. But the defender of the ontological argument can take no comfort from such an observation, since it defends Premise 2 by using the conclusion of the ontological argument.
- There is still room for the work in Oppenheimer and Zalta 1991, since the earlier paper offers an independent argument (based on Premise 1) that the description denotes, and preserves the ontological argument based on a revised Premise 2: $\neg E!x \rightarrow \exists y(Gyx \mathbin{\&} Cy)$.

# Bibliography


- Fitelson, B. and E. Zalta, 2007, 'Steps Towards a Computational Metaphysics', *Journal of Philosophical Logic*, 36/2 (April): 227–247.
- Lewis, D., 1986, *On the Plurality of Worlds*, Oxford: Blackwell.
- Oppenheimer, P., and E. Zalta, 1991, 'On the Logic of the Ontological Argument', *Philosophical Perspectives*, 5: 509–529; selected for republication in *The Philosopher's Annual: 1991*, Volume XIV (1993): 255–275.
- Pelletier, F.J. and E. Zalta, 2000, 'How to Say Goodbye to the Third Man', *Noûs*, 34/2 (June): 165–202.
- Russell, B., 1905, 'On Denoting',
- Zalta, E., 1993, 'Twenty-Five Basic Theorems in Situation and World Theory', *Journal of Philosophical Logic*, 22: 385–428;
- Zalta, E., 2000, 'A (Leibnizian) Theory of Concepts', *Philosophiegeschichte und logische Analyse / Logical Analysis and History of Philosophy*, 3: 137–183.