Analyses of Analysis - Part I - Exegetical Analysis

Roger Bishop Jones

Created 2009/06/18

Last Change Date: 2011/01/05 11:02:31

Id: b001.tex,v 1.13 2011/01/05 11:02:31 rbj Exp

Contents

1 Introduction **9**
1.1 Methods . . . 9
1.2 Themes . . . 9

2 Plato and Aristotle **11**
2.0.1 Preliminary Formalities . . . 12
2.1 Metaphysics (I) . . . 13
2.1.1 The Grice/Code/Speranza Formulae . . . 13
2.1.2 References to Plato . . . 13
2.1.3 Aristotelian References . . . 14
2.1.4 Formal Principles . . . 14
2.1.5 Total Definitions . . . 20
2.1.6 Partial Definitions . . . 22
2.1.7 Ontological Theorems . . . 23
2.1.8 Platonic Principles and Theorems . . . 24
2.1.9 Some Comments on The Conjectures . . . 25
2.2 The Organon . . . 27
2.2.1 Models and Their Significance . . . 28
2.2.2 Preliminaries . . . 31
2.2.3 Interpretation in Set Theory . . . 39
2.2.4 Propositional Interpretation . . . 43
2.2.5 Naive Interpretation in Predicate Calculus . . . 45
2.2.6 Predicate Calculus Without Empty Terms . . . 50
2.2.7 Existential Import in Universals . . . 54
2.2.8 Existential Import in Affirmations . . . 59
2.2.9 Modal Syllogisms . . . 63
2.2.10 Demonstrative Truth . . . 71
2.3 Metaphysics (II) . . . 72
2.3.1 Semantics . . . 73
2.3.2 Predication . . . 76
2.3.3 Propositional Operators . . . 79
2.3.4 Quantification . . . 80
2.3.5 Judgements . . . 81

2.3.6 Conversions . . . 82
2.3.7 Modal Conversions . . . 83
2.3.8 Other Conversions . . . 84
2.3.9 Syllogisms for Essential Predication . . . 85
2.3.10 Some Accidental Syllogisms . . . 85
2.3.11 Grice and Code . . . 85
2.4 Conclusions . . . 88

3 Leibniz **91**
3.1 Leibniz On Identity . . . 92
3.2 The Calculus Ratiocinator . . . 94
3.2.1 Leibniz's Interpretation of Aristotle's Syllogistic . . . 94
3.3 Metaphysics . . . 99
3.3.1 The Subject Matter . . . 100
3.3.2 Predication . . . 101
3.3.3 Propositional Operators . . . 103
3.3.4 Quantification . . . 104
3.3.5 Judgements . . . 105
3.3.6 Conversions . . . 106
3.3.7 Modal Conversions . . . 107
3.3.8 Other Conversions . . . 108
3.3.9 Syllogisms . . . 109

4 Hume and Kant **111**

5 Frege **113**

6 Russell and Wittgenstein **115**
6.1 Introduction . . . 116
6.1.1 Preliminary Ideas . . . 116
6.2 Notes on Russell . . . 117
6.2.1 Principia Mathematica . . . 117
6.2.2 Part I . . . 123
6.3 Preliminary Observations . . . 123
6.3.1 Schematic Variables . . . 124
6.4 The Tractatus . . . 125
6.4.1 The World . . . 125
6.4.2 States of Affairs . . . 126
6.4.3 Thoughts and Propositions . . . 126
6.4.4 Propositions as Truth Functions - I . . . 126
6.4.5 Propositions . . . 127

7 Grice **129**
7.0.6 Background . . . 130

7.0.7 Preliminary Formalities . . . 131
7.1 The Problem . . . 131
7.2 System Q: Objectives . . . 132
7.3 Classical Logic using HOL . . . 133
7.3.1 Scope . . . 133
7.3.2 Semantic Domains . . . 135
7.3.3 Descriptions . . . 136
7.3.4 Propositional Logic . . . 137
7.3.5 Quantifier Inference-Rules . . . 138
7.3.6 P Committal . . . 138
7.4 System C . . . 139
7.4.1 Quantifier Inference-Rules . . . 142
7.4.2 Existence . . . 143
7.4.3 Identity . . . 144
7.4.4 Names and Descriptions . . . 145
7.5 System S . . . 146
7.5.1 Quantifier Definitions . . . 151
7.5.2 Quantifier Inference-Rules . . . 151
7.5.3 Existence . . . 152
7.5.4 Identity . . . 153
7.6 Conclusions . . . 155

8 Conclusions **157**

Bibliography **158**

Index **158**

Chapter 1

Introduction

1.1 Methods

The analyses in this volume of aspects of the work of various important philosophers are undertaken using modern formal methods with the help of supporting information technology. Information about these tools and methods is presented in Volume II of this work, which is concerned with synthetic rather than exegetical analysis.

1.2 Themes

It is my aim to trace the history and origin of some of the ideas which underpin formal analysis of the kinds addressed in Volume II. It is to be expected that as the work progresses my ideas about what the important ideas are will evolve, and this account of the principle ideas will be expanded and refined.

In the first place the following short list will suffice:

- The distinction between logical and factual truths. My use of the definite article here reflects my prejudice that there is one important distinction which has and continues to evolve corresponding to Hume's distinction between ¡i¿relations between ideas¡/i¿ and ¡i¿matters of fact¡/i¿. The various related dichotomies include:
  - Necessity and Contingency
  - Essential and Accidental predication
  - Demonstrative truth
  - a priori and a posteriori knowledge
  - the analytic/synthetic distinction
  - Logical verses empirical truth.

  - The notions of tautology, truth functions and truth conditions

- The evolution of formality.

Chapter 2

Plato and Aristotle

My purpose here is to use formal models to aid in understanding the philosophies of Plato and Aristotle, both in relation to their contribution generally to the areas of interest, philosophical logic, semantics and metaphysics, and also more specifically in relation to the extent to which these philosophers laid the ground for the distinction which was later expressed in Hume's fork.

In doing this I began with some enquiries into Aristotle's metaphysics published by Code [**?**] and produced from this a preliminary model (Section 2.1). In these an important defect is that the model does not support the u-p syllogisms on which Code's analysis depends more heavily than one might have expected, and also does not allow for modal operators, which not only enter into Code's material but are also important for the kinds of comparison with later philosophers which I had hoped to undertake. I have also failed at this stage to bring out the distinction between Plato and Aristotle. Perhaps more important is that I did not arrive at a good understanding of Code and the model is therefore unlikely to fully reflect his intensions. It is also the case that the method I adopted for the analysis of Code is one which he would have been likely to question. His paper does briefly discuss unfavourably the interpretation of Aristotle in terms of modern idom such as that of set theory. It is an important part of my objectives in this document to discuss this kind of formal exgetics, and hopefully to explain contra Code why the use of modern set theoretic language is appropriate and helpful in the analysis of materials which could not have been originally conceived in such terms.

I then went back from the Metaphysics to the Organon and used formal models to come to a better understanding of Aristotle's formal syllogistic logic (Section 2.2). In this seven models of increasing sophistication were produced and formed the basis for undertaking a further model of the metaphysics which incorporated the u-p syllogisms and modal operators (Section 2.3). All of this is preliminary to addressing the real issues, which has not yet seriously begun.

2.0.1 Preliminary Formalities

In the document several different formal models are presented. By and large they are independent, but a some features are common and are therefore presented here for use in all the models.

SML

```
open_theory "misc2";
force_new_theory "aristotle";
```

We define inequality:

SML

```
declare_infix (300, "≠");
```

HOL Constant

$$\$\neq : 'a \to 'a \to BOOL$$

$$\forall x\ y \bullet\ x \neq y \Leftrightarrow \neg\ x = y$$

2.1 Metaphysics (I)

In this section we consider some material on Aristotle's Metaphysics [**?**] which originated in work of Grice and Code [**?**] and came to me from a posting of J.L. Speranza on the hist-analytic mailing list. Code's paper is also partially available at Google Books.

What Speranza posted was the list of formulae which are named below as c01 through c31 (though not exactly as given, I have massaged them to be acceptable to HOL and also have quantified over all free variables).

The analysis in this section is independent of the preceding analysis of Aristotle's syllogism, and considers predication from a rather different point of view, which hangs around the distinction between essential and accidental predication. In the next section I will produce another model in which essence and accident are combined with a full treatment of modal syllogism so that some conclusions might be drawn about the relationship between essence and necessity in Aristotelean philosophy.

A new theory is needed which I will call "ariscat" which is created here:

SML

```
open_theory "aristotle";
force_new_theory "ariscat";
```

2.1.1 The Grice/Code/Speranza Formulae

This work began with an attempt to analyse using ProofPower a set of formulae posted by J.L. Speranza to the hist-analytic mailing list. These were a Speranzan transcript of formulae published in a paper by Code [**?**], the work presented in that paper having begun with some joint work with H.P. Grice [**?**].

The following material labelled "Code/Speranza" began as a transcription from Speranza's email, and was later updated when Speranza pointed me to the partial availability of the Code paper on Google Books. I then put back some of the detail missed in the Speranza version, enclosed in square brackets, and enclosed in curly braces some of the material which Speranza had added to Code's.

The terminology used is Grice's. Code uses "Is" and "Has" instead of "izz" and "hazz" (which were coined by Grice and used by Speranza). Aristotle's originals have been translated as "SAID OF" and "IN", according to Cohen (Grice and Code on IZZing and HAZZing).

The material is interspersed with a formalisation in ProofPower-HOL. I have adopted some of Code's headings for sections.

2.1.2 References to Plato

By footnote in Code.

30 *Phaedo*. $102B8C - 1$ with C10ff (cf. B 4-6)

2.1.3 Aristotelian References

These are gathered together temporarily and will later be distributed as footnotes.

Cat. 1^a**, 12–15** Equivocally, Univocally and Derivatively.

Cat. 1^b5**, 3b12**

Cat. 2^a**, 19–34** Presence and Predication.

Cat. 3^a**, 15–20** Subclasses and Predicability.

Cat. 10^a**, 27ff.** Opposite, Contrary, Privative, Positive.

De. Int. **7,** 17^a**9–40** Universal and Individual Subjects.

*Post. An. A*4, 73^a**34**–$^b5$ What are the premisses of demonstration. Distinguishing essential and accidental predication.

*Metap. I*1, 1059^a10-14 The One.

Metap. Δ9, 1018^a1-4 Same and Different. 'Socrates' and 'musical Socrates' are thought to be the same; but 'Socrates' is not predicable of more than one subject, and therefore we do not say 'every Socrates' as we say 'every man'.

Metap. Δ18, 1022^a25-27 In Virtue Of. Admissibility of self-predication of particulars.

Metap. Δ23, 1023^a11-13 Having and Being. Two kinds of predication.

*Metap. Z*5 Only substance is definable.

Metap. $Z6, 1032^a4-6$ Each thing and its essence are one and the same.

Metap. $Z8, 1034^a7$ different in virtue of their matter

Metap. $Z11, 1037^a33-^b4$ Formulae, Parts, Substance and Essence.

Topics. $H(VII)1, 152^b25-29$ Numerical identity.

2.1.4 Formal Principles

We begin with "Formal Principles" which we take as implicit definitions of two kinds of Aristotelian predication.

Code/Speranza

```
[(A) Formal Principles]

[FP1] 1. A izz A.
[FP2] 2. (A izz B & B izz C) −−> A izz C.
[FP3] 3. A hazz B −> −(A izz B).
[FP4] 4. A hazz B iff A hazz Some−Thing [something] that izz B.
```

The modelling in ProofPower-HOLwill be entirely conservative, so we provide explicit definitions for "izz" and "hazz" and prove that they satisfy these principles and suffice also for the definitions and theorems which follow.

If the formal development is complete the definitions will have been shown to be sufficient. In order to test whether the principles suffice I will attempt to proceed on these alone, though I suspect that will not be possible. Failing that I will offer informal arguments to the contrary (a formal argument would require ascent to a metatheory which would involve too much work).

In order to define these concepts we have to decide what they are about, and this is not straightforward.

Categories

Aristotle has a system of categories, and these seem central to the topic. Much hangs on what these are, and to get a nice structure to our theory it seems advisable to do a bit of "category theory" first. Of course this is not at all the same thing as the branch of mathematics which now goes by that name, but the choice of name for the mathematics was not entirely quixotic and at some point it might be interesting to think about the relationship between the two kinds of category theory.

Among these categories that of substances plays a special role. Substances can be particular in which case they correspond to some individual, or not, in which case they are sets of individuals. The particulars of the other categories are attributes, and the non-individuals are sets of attributes. I don't think you can have singleton sets, so we can model all these categories as sets of sets in which the singleton sets are the individuals. Attributes can also be considered as sets of individual substances and so there is a type difference between the category of substances and the other categories.

The following introduce new types and type abbreviations for modelling Aristotle's categories.

ACAT is a type of attribute categories

ISUB is a type of individual substances

CATM is the type of the things which are in categories.

> This is a 'disjoint union', which means that there are two kinds of thing which one finds in categories, either a set of individual substances (using singleton sets to represent individual substances), or a set of properties of individual substances tagged with an attribute category.

CAT is a type abbreviation for a notion of category which is either an attribute category or some other category (which will stand for the category of substances).

There are some oversimplifications here which I am hoping will not be too serious for a useful first cut.

Modal Operators The main one is that this general approach will not permit the definition of modal operators, which are used by Code. Whether there is real need for them seems to me doubtful, but if necessary a modal model could be provided.

Empty Sets The second is that I have not excluded empty sets, and hence that there can be predicates with null extensions. This is in fact consistent with Code's "principles", but we find that his definitions are written as if there were no empty predicates, even though this is not entailed by the principles. Where Code's definitions presume non-emptiness of predicates I have chosen another definition which does not. Some of the theorems are then unprovable.

Extensionality Intracategorial equality will be extensional. That appears to be what is required, so it probably isn't a problem!

Predicability Aristotle defines particulars in terms of Predicability, they are the impredicables. Its not clear how to deal with this, and it appears to be in terminological conflict with Code, who appears to use use "predicable" to mean "truly predicable". Code has the general principle "A izz A" which implicitly asserts that everything is truly predicable of itself.

SML

$new\_type(\texttt{"}\boldsymbol{ACAT}\texttt{"},\ 0);$
$new\_type(\texttt{"}\boldsymbol{ISUB}\texttt{"},\ 0);$
$declare\_type\_abbrev\ (\texttt{"}\boldsymbol{CATM}\texttt{"},\ [],\ \ulcorner{:}ISUB\ \mathbb{P} + (ACAT \times (ISUB \to BOOL)\mathbb{P})\urcorner);$
$declare\_type\_abbrev\ (\texttt{"}\boldsymbol{CAT}\texttt{"},\ [],\ \ulcorner{:}ONE + ACAT\urcorner);$

We name the category of substances.

HOL Constant

$\boldsymbol{CatSubs} : CAT$

$CatSubs = InL\ One$

Now we define various operators over categories and their constituents which suffice for the development of an appropriate theory, in the context of which rest of the Aristotelian terminology will we hope prove definable.

First "projection" functions which yield the constituents of $MCAT$s.

HOL Constant

$$\boldsymbol{Cat} : CATM \to CAT$$

$$\forall x \bullet\ Cat\ x = if\ IsL\ x\ then\ CatSubs\ else\ InR(Fst\ (OutR\ x))$$

HOL Constant

$$\boldsymbol{IndvSet} : CATM \to ISUB\ \mathbb{P}$$

$$\forall x \bullet\ IndvSet\ x = OutL\ x$$

HOL Constant

$$\boldsymbol{AttrSet} : CATM \to (ISUB \to BOOL)\mathbb{P}$$

$$\forall x \bullet\ AttrSet\ x = Snd(OutR\ x)$$

This one turns out handy.

HOL Constant

$$\boldsymbol{CatSet} : CATM \to (ISUB + (ISUB \to BOOL))\mathbb{P}$$

$$\begin{array}{l}\forall x \bullet\ CatSet\ x = \\ \quad if\ Cat\ x = CatSubs \\ \quad then\ \{y \mid \exists z \bullet\ z \in IndvSet\ x \wedge y = InL\ z\} \\ \quad else\ \{y \mid \exists z \bullet\ z \in AttrSet\ x \wedge y = InR\ z\}\end{array}$$

With these definitions in place we get a useful characterisation of identity for elements of $CATM$.

$$\begin{array}{l}catm\_eq\_lemma = \\ \quad \vdash \forall\ A\ B \bullet\ A = B \Leftrightarrow Cat\ A = Cat\ B \wedge CatSet\ A = CatSet\ B\end{array}$$

Predication

Now we can define predication. We do this in terms of Grice's *izz* and *hazz*.

SML

```
declare_infix (300, "izz");
declare_infix (300, "hazz");
```

HOL Constant

$$\$\boldsymbol{izz} : CATM \to CATM \to BOOL$$

$$\forall A\ B \bullet A\ izz\ B \Leftrightarrow Cat\ A = Cat\ B \wedge CatSet\ A \subseteq CatSet\ B$$

HOL Constant

$$\$\boldsymbol{hazz} : CATM \to CATM \to BOOL$$

$$\forall A\ B \bullet A\ hazz\ B \Leftrightarrow Cat\ A = CatSubs \wedge \neg\ Cat\ B = CatSubs \wedge \exists a \bullet\ a \in AttrSet\ B \wedge \forall s \bullet\ s \in IndvSet\ A \Rightarrow a\ s$$

That was reasonably neat, but the definition of *izz* isn't terribly convenient for proving things. Lets have some *izz* lemmas:

$$izz\_lemma1 = \vdash \forall\ A\ B \bullet\ Cat\ A = CatSubs \Rightarrow (A\ izz\ B \Leftrightarrow Cat\ B = CatSubs \wedge IndvSet\ A \subseteq IndvSet\ B)$$

$$izz\_lemma2 = \vdash \forall\ A\ B \bullet\ Cat\ B = CatSubs \Rightarrow (A\ izz\ B \Leftrightarrow Cat\ A = CatSubs \wedge IndvSet\ A \subseteq IndvSet\ B)$$

$$izz\_lemma3 = \vdash \forall\ A\ B \bullet \neg\ Cat\ A = CatSubs \Rightarrow (A\ izz\ B \Leftrightarrow Cat\ B = Cat\ A \wedge AttrSet\ A \subseteq AttrSet\ B)$$

$$izz\_lemma4 = \vdash \forall\ A\ B \bullet \neg\ Cat\ B = CatSubs \Rightarrow (A\ izz\ B \Leftrightarrow Cat\ B = Cat\ A \wedge AttrSet\ A \subseteq AttrSet\ B)$$

The Principles in HOL

Here are the HOL versions of the Code "Principles".

The following is a bit of program in a programming language called SML, which stands for "Standard Meta Language"! It names various terms in HOL, the name on the left 'c01' (short for 'conjecture 1'), the term on the right quoted in "Quine corners". [1]

[1] "Quine corners" are a notation originally used by Quine for Godel numbers, i.e., in Quine's use '⌜43⌝' is a friendly way of writing doen the Godel number of the numeral '43'. In ProofPower HOL these corners are used to refer to HOL terms in the metalanguage SML. In HOL, a formula is a term of type ⌜:BOOL⌝ (the opening ⌜: is used when quoting a type rather than a term).

SML

```
val c01 = ⌜∀A• A izz A⌝;
val c02 = ⌜∀A B C• A izz B ∧ B izz C ⇒ A izz C⌝;
val c03 = ⌜∀A B• A hazz B ⇒ ¬ A izz B⌝;
val c04 = ⌜∀A B• A hazz B ⇔ ∃C• A hazz C ∧ C izz B⌝;
```

One would expect this set of principles to be sufficient to characterise *izz* and *hazz* (i.e. sufficient to derive any other true facts about them) but this seems doubtful.

Here are some supplementary conjectures.

SML

```
val c01b = ⌜∀A• ¬ A hazz A⌝;
val c03b = ⌜∀A B• A izz B ⇒ ¬ A hazz B⌝;
val c03c = ⌜∀A B• A hazz B ⇒ ¬ A = B⌝;
val c04a = ⌜∀A B C• A hazz B ∧ B izz C ⇒ A hazz C⌝;
val c04b = ⌜∀A B C• A izz B ∧ B hazz C ⇒ A hazz C⌝;
```

Of the supplementaries:

- c01b is derivable from c03 and c01.
- c03b is the contrapositive of c03.
- c03c would be derivable for Code once he has defined equality, it is provable for us now because we have a primitive equality.
- c04a is a preferable formulation of the right-left implication in c04, and we have used it to prove c04.
- c04b is an obvious further transitivity-like property, which does not look like it's provable from the stipulated principles.

Proven Theorems

```
l01  = ⊢ ∀ A• A izz A
l01b = ⊢ ∀ A• ¬ A hazz A
l02  = ⊢ ∀ A B C• A izz B ∧ B izz C ⇒ A izz C
l03  = ⊢ ∀ A B• A hazz B ⇒ ¬ A izz B
l03b = ⊢ ∀ A B• A izz B ⇒ ¬ A hazz B
l03c = ⊢ ∀ A B• A hazz B ⇒ ¬ A = B
l04a = ⊢ ∀ A B C• A hazz B ∧ B izz C ⇒ A hazz C
l04  = ⊢ ∀ A B• A hazz B ⇔ (∃ C• A hazz C ∧ C izz B)
l04b = ⊢ ∀ A B C• A izz B ∧ B hazz C ⇒ A hazz C
```

2.1.5 Total Definitions

Code/Speranza

```
[(B) Total Definitions]

{6. (A hazz B & A is a particular) -> there is a C such that (C =/= A) & (A izz B).}

[D1] 7. A is predicable of B iff ((B izz A) ∨ (B hazz Something that izz A).
[D2] 8. A is essentially predicable [L−predicable] of B iff B izz A.
[D3] 9. A is accidentally predicable [H−predicable] of B iff B hazz something that izz A.
[D4] 10. A = B iff A izz B & B izz A.
[D5] 11. A is an individual iff (Nec)(For all B) B izz A -> A izz B
[D6] 12. A is a particular iff (Nec)(For all B) A is predicable of B -> (A izz B & B izz A)
[D7] 13. A is a universal iff
            (Poss) (There is a B) A is predicable of A[B] & −(A izz B & B izz A)
```

There is a certain amount of duplication of terminology here, since essential and accidental predication seem to be just *izz* and *hazz* backwards. I'm not so happy with the "ables" here, for what is clearly meant is "truly predicable", which is not quite the same thing. Better names would be simply "is_essentially" and "is_accidentally", lacking the ambiguity of "able" (but then they would have to be the other way round, exactly the same as *izz* and *hazz*).

Anyway here are the definitions (keeping the names (more or less) as they were for the present):

SML

```
declare_infix (300, "predicable_of");
declare_infix (300, "essentially_predicable_of");
declare_infix (300, "accidentally_predicable_of");
```

HOL Constant

```
$essentially_predicable_of : CATM → CATM → BOOL
```

```
∀A B• A essentially_predicable_of B ⇔ B izz A
```

HOL Constant

```
$accidentally_predicable_of : CATM → CATM → BOOL
```

```
∀A B• A accidentally_predicable_of B ⇔ B hazz A
```

Aristotelian predication is then:

HOL Constant

$$\$\textbf{\textit{predicable\_of}} : CATM \to CATM \to BOOL$$

$$\forall A\ B\bullet\ A\ predicable\_of\ B \Leftrightarrow A\ essentially\_predicable\_of\ B \vee A\ accidentally\_predicable\_of\ B$$

Because we have not precluded empty predicates Code's definition will not do, and we have to make "individual" primitive, insisting on an individual being a singleton.

HOL Constant

$$\textbf{\textit{individual}} : CATM \to BOOL$$

$$\forall A\bullet\ individual\ A \Leftrightarrow \exists a\bullet\ CatSet\ A = \{a\}$$

According to Code's definition a particular is a substantial individual, we also have to use a more direct statemant of that principle.

HOL Constant

$$\textbf{\textit{particular}} : CATM \to BOOL$$

$$\forall A\bullet\ particular\ A \Leftrightarrow individual\ A \wedge Cat\ A = CatSubs$$

Again we have a problem with Code's definition and therefore define a universal as a non-particular.

HOL Constant

$$\textbf{\textit{universal}} : CATM \to BOOL$$

$$\forall A\bullet\ universal\ A \Leftrightarrow \neg\ particular\ A$$

SML

$$val\ c06 = \ulcorner\forall A\ B\bullet\ A\ hazz\ B \wedge particular\ A \Rightarrow \exists C\bullet\ C \neq A \wedge A\ izz\ B\urcorner;$$
$$val\ c06n = \ulcorner\neg\ \forall A\ B\bullet\ A\ hazz\ B \wedge particular\ A \Rightarrow \exists C\bullet\ C \neq A \wedge A\ izz\ B\urcorner;$$
$$val\ c07 = \ulcorner\forall A\ B\bullet\ A\ predicable\_of\ B \Leftrightarrow (B\ izz\ A) \vee \exists C\bullet\ B\ hazz\ C \wedge C\ izz\ A\urcorner;$$
$$val\ c08 = \ulcorner\forall A\bullet\ A\ essentially\_predicable\_of\ B \Leftrightarrow B\ izz\ A\urcorner;$$
$$val\ c09 = \ulcorner\forall A\bullet\ A\ accidentally\_predicable\_of\ B \Leftrightarrow \exists C\bullet\ B\ hazz\ C \wedge C\ izz\ A\urcorner;$$
$$val\ c10 = \ulcorner\forall A\ B\bullet\ A = B \Leftrightarrow A\ izz\ B \wedge B\ izz\ A\urcorner;$$
$$val\ c11 = \ulcorner\forall A\ B\bullet\ individual\ A \Leftrightarrow \natural(\forall B\bullet\ B\ izz\ A \Rightarrow A\ izz\ B)\urcorner;$$
$$val\ c12 = \ulcorner\forall A\bullet\ particular\ A \Leftrightarrow \natural(\forall B\bullet\ A\ predicable\_of\ B \Rightarrow A\ izz\ B \wedge B\ izz\ A)\urcorner;$$
$$val\ c13 = \ulcorner\forall A\bullet\ universal\ A \Leftrightarrow \Diamond(\exists B\bullet\ (A\ predicable\_of\ B \wedge \neg(A\ izz\ B \wedge B\ izz\ A)))\urcorner;$$

$l05 \quad = \vdash \forall\ x\bullet\ form\ x \Rightarrow universal\ x$
$l06n \quad = \vdash \neg\ (\forall\ A\ B\bullet\ A\ hazz\ B \wedge particular\ A \Rightarrow (\exists\ C\bullet\ C \neq A \wedge A\ izz\ B))$
$l07 \quad = \vdash \forall\ A\ B\bullet\ A\ predicable\_of\ B \Leftrightarrow B\ izz\ A \vee (\exists\ C\bullet\ B\ hazz\ C \wedge C\ izz\ A)$
$l08 \quad = \vdash \forall\ A\bullet\ A\ essentially\_predicable\_of\ B \Leftrightarrow B\ izz\ A$
$l09 \quad = \vdash \forall\ A\bullet\ A\ accidentally\_predicable\_of\ B \Leftrightarrow (\exists\ C\bullet\ B\ hazz\ C \wedge C\ izz\ A)$
$l10 \quad = \vdash \forall\ A\ B\bullet\ A = B \Leftrightarrow A\ izz\ B \wedge B\ izz\ A$

c06 is false (see l06n), probably a typo. However, I couldn't work out what was intended.

c11-13 are not provable in our model because of the existence of empty predicates (and the lack of modal operators).

2.1.6 Partial Definitions

Code/Speranza

$[(C)\ Partial\ Definitions]$

$[D8]\ 14.\ If\ A\ is\ Some\ Thing\ [a\ this\ somewhat],\ A\ is\ an\ \ individual.$
$[D9]\ 15.\ If\ A\ is\ a\ [(seperable)\ Platonic]\ Form,$
$\qquad\qquad A\ is\ Some\ Thing\ [a\ this\ somewhat]\ and\ Universal.$

This is D8/c14.

HOL Constant

$$\boldsymbol{SomeThing} : CATM \rightarrow BOOL$$

$$\forall x\bullet\ SomeThing\ x \Rightarrow individual\ x$$

A form is a non-substantial individual.

This is D9/c15

HOL Constant

$$\boldsymbol{form} : CATM \rightarrow BOOL$$

$$\forall x\bullet\ form\ x \Rightarrow SomeThing\ x \wedge universal\ x$$

2.1.7 Ontological Theorems

Code/Speranza

```
[(D) Ontological Theorems]

[T1] 16. A is predicable of B iff (B izz A) v (B hazz Some Thing that Izz A).
[T2] 17. A is essentially predicable [L−predicable] of A.
[T3] 18. A is accidentally predicable [H−predicable] of B −>  A =/= B
[T4] A is not accidentally predicable [H−predicable] of A
{19. − (A is accidentally predicable of B) −> A =/= B.}
[T5] 20. A is a particular −> A is an individual.
      [Note that the converse of T5 is not a theorem]
[T6] 21. A is a particular −> No Thing [nothing] that is Not Identical with A izz A.
[T7] 22. No Thing is both particular & a [(separable) Platonic] Form.
[T8] 23. A is a (seperable Platonic) Form −> nothing that is not identical with A izz A.
[T9] 24. A is a particular −> there is no (seperable Platonic) form B such that A izz B.
[T10] 25. A is a (seperable Platonic) form
           −> ((A is predicable of B & A =/= B) −> B hazz A)
[T11] 26. (A is a (seperable Platonic) form & B is a particular)
           −> (A is predicable of B iff B hazz A).
```

SML

```
val c05 = ⌜∀x• universal x ⇒ form x⌝;
val c05b = ⌜∀x• form x ⇒ universal x⌝;
val c16 = ⌜∀A B• A predicable_of B ⇔ (B izz A) ∨ ∃C• (B hazz C ∧ C izz A)⌝;
val c17 = ⌜∀A• A essentially_predicable_of A⌝;
val c18 = ⌜∀A• A accidentally_predicable_of B ⇒ A ≠ B⌝;
val c19 = ⌜∀A• ¬ A accidentally_predicable_of A⌝;
val c20 = ⌜∀A• particular A ⇒ individual A⌝;
val c21 = ⌜∀A• particular A ⇒ ¬ ∃C• C ≠ A ∧ C izz A⌝;
val c22 = ⌜¬ ∃A• particular A ∧ form A⌝;
val c23 = ⌜∀A• form A ⇒ ¬ ∃C• C ≠ A ∧ C izz A⌝;
val c23b = ⌜∀A• form A ⇒ individual A⌝;
val c24a = ⌜∀ A B• particular A ∧ individual B ∧ A izz B ⇒ particular B⌝;
val c24 = ⌜∀A• particular A ⇒ ¬ ∃B• form B ∧ A izz B⌝;
val c24b = ⌜∀A• particular A ⇒ ¬ form A⌝;
val c25 = ⌜∀A B• form A ⇒ A predicable_of B ∧ A ≠ B ⇒ B hazz A⌝;
val c26 = ⌜∀A B• form A ∧ particular B ⇒ (A predicable_of B ⇔ B hazz A)⌝;
```

These are the ones I have proved.

$l16 \quad = \vdash \forall\ A\ B\bullet\ A\ predicable\_of\ B \Leftrightarrow B\ izz\ A \vee (\exists\ C\bullet\ B\ hazz\ C \wedge C\ izz\ A)$
$l17 \quad = \vdash \forall\ A\bullet\ A\ essentially\_predicable\_of\ A$
$l19 \quad = \vdash \forall\ A\bullet \neg\ A\ accidentally\_predicable\_of\ A$
$l20 \quad = \vdash \forall\ A\bullet\ individual\ A \Rightarrow particular\ A$
$l22 \quad = \vdash \neg\ (\exists\ A\bullet\ particular\ A \wedge form\ A)$
$l23b \quad = \vdash \forall\ A\bullet\ form\ A \Rightarrow individual\ A$
$l24a \quad = \vdash \forall\ A\ B\bullet\ particular\ A \wedge individual\ B \wedge A\ izz\ B \Rightarrow particular\ B$
$l24 \quad = \vdash \forall\ A\bullet\ particular\ A \Rightarrow \neg\ (\exists\ B\bullet\ form\ B \wedge A\ izz\ B)$
$l26 \quad = \vdash \forall\ A\ B\bullet\ form\ A \wedge particular\ B \Rightarrow (A\ predicable\_of\ B \Leftrightarrow B\ hazz\ A)$

T6/c21, T8/c23, T10/c25 are all unprovable because of the existence of empty predicates.

2.1.8 Platonic Principles and Theorems

This section is a bit of a mess. I now see that the reason for this is that Code is now presenting a different theory here, which is Aristotle's conception of Plato's metaphysics. This explains why these principles at least augment (and possibly contradict) concepts which have already been defined. In our method, which involves, for the sake of ensuring consistency, the use of only conservative extensions, this cannot be done simply by adding new principles. We have to develop two systems in separate theories in which the differences of conception between Aristotle and Aristotle's conception of Plato are investigated in distinct contexts (though we could place in a single parent theory the elements which are common to both).

This will be considered later.

Code/Speranza

[(E) *Platonic Principle*]

[*PP1*] *5. Each universal is a (seperable Platonic) form.*
[*PP2*] *27. (A is particular & B is a universal & predicable of A)*
−> there is a C such that (A =/= C & C is essentially predicable of A)

SML

$val\ c05 = \ulcorner \forall x\bullet\ universal\ x \Rightarrow form\ x \urcorner;$
$val\ c05b = \ulcorner \forall x\bullet\ form\ x \Rightarrow universal\ x \urcorner;$
$val\ c27 = \ulcorner \forall A\ B\bullet\ particular\ A \wedge universal\ B \wedge B\ predicable\_of\ A$
$\Rightarrow \exists C\bullet\ (A \neq C \wedge C\ essentially\_predicable\_of\ A) \urcorner;$

c05 is not provable, its converse c05b is. c27 is not provable, since it would require that there be more than one particular and we have no reason to believe that to be the case.

$l05b \quad = \vdash \forall\ x\bullet\ form\ x \Rightarrow universal\ x$

Code/Speranza

$[(F)\ Platonic\ Theorem]$

$\{28.\ If\ there\ are\ particulars,\ of\ which\ universals\ are\ predicable,$
$\quad not\ every\ universal\ is\ Some\ Thing.\}$
$[PT1]\ 29.\ Each\ universal\ is\ Some\ Thing\ [a\ this\ somewhat].$
$[PT2]\ 30.\ If\ A\ is\ a\ particular,\ there\ is\ no\ B\ such\ that$
$\quad (A =/= B\ \&\ \ B\ is\ essentially\ predicable\ of\ A).$
$[PT3]\ 31.\ (A\ is\ predicable\ of\ B\ \&\ A =/= B) -> A\ is\ accidentally\ predicable\ of\ B.$

SML

$val\ c28 = \ulcorner(\exists P\bullet\ particular\ P \wedge \exists U\bullet\ universal\ U \wedge U\ predicable\_of\ P)$
$\quad \Rightarrow \neg\ (\forall U\bullet\ universal\ U \Rightarrow thing\ U)\urcorner;$
$val\ c29 = \ulcorner\forall U\bullet\ universal\ U \Rightarrow thing\ U\urcorner;$
$val\ c30 = \ulcorner\forall A\bullet\ particular\ A \Rightarrow \neg\ \exists B\bullet\ (A \neq B \wedge B\ essentially\_predicable\_of\ A)\urcorner;$
$val\ c31 = \ulcorner\forall A\ B\bullet\ A\ predicable\_of\ B \wedge A \neq B \Rightarrow A\ accidentally\_predicable\_of\ B\urcorner;$

These are the ones I have proved.

$l06n = \quad \vdash \neg\ (\forall\ A\ B\bullet\ A\ hazz\ B \wedge particular\ A \Rightarrow (\exists\ C\bullet\ C \neq A \wedge A\ izz\ B))$

2.1.9 Some Comments on The Conjectures

The main problem with the conjectures is that as a group they are inconsistent. Consequently, one cannot find definitions which are consistent with all the conjectures.

So first I will expose some of the most obvious contradications which flow from the conjectures.

1. From 5 and 15 we conclude that form and universal are coextensive. I think it may be that this is part of the Platonic view but not of the Aristotelian one.

Here are some observations on specific conjectures (now out of date).

c01 Note here that this is quantified over everything, and hence over individuals, whereas Aristotle describes individuals as thing of which one may predicate, but which are not themselves predicable. Perhaps this inderdiction applies to *hazz* but not to *izz*, to accidental but not essential predication. (We should add the rule ⌜¬ A hazz A⌝ which is easy to prove.) I don't know any

more detail about Aristotle's attitude towards predication by individuals. If one cannot, where do we stand when we do, as in "Socrates is Socrates" and "Socrates is Aristotle". Anyway, if these were to have a truth value (which surely they do) then the truth value will be as in this rule.

c02 Behind the scenes this is transitivity of set inclusion.

c03 This is because *izz* is intracategorial and *hazz* is intercategorial. An obvious but useful corrolary is that they are not equal (c03b).

c04 This, and most of the other theorems involving existential quantification, is rather odd. Its proof depends on the conjecture $c04a$, which we have proven, and which involves no existential quantification, but its content is not significantly greater than that rule. From right to left $l04$ is $l04b$ (you just pull out the existential and it turns into a universal). From left to right $l04$ is trivial, since B serves as a witness for the existential.

Ideally we would be working with claims which are expressible syllogistically, i.e. without benefit of quantifiers. We can make an exception for universals on the left, since these are interconvertible with the conjecture with free variables instead which we can think of as schemata. Where an existential quantifier appears in a negative context it will turn universal if pulled out to the top level and can therefore be dispensed with. Elsewhere its worth asking whether the content is significant (including, as here, one half of the content implicit in putting an existential under an equivalence).

$co4b$ is an obvious similar result to $c04a$.

c05 According to my definition this is the wrong way round.

c06 As it stands this is provably false since we have $\ulcorner AhazzB \urcorner$ on the left, which entails that A and B are not of the same category, and $\ulcorner AizzB \urcorner$ in the right, which entails that they are of the same category.

c07 This, at the expense of using an existential, nevertheless tells us nothing that is not immediate from the definitions.

c08 Is just our definition.

c09 This turns into $c04$ (apart from the variable names) once you expand the definition of *accidentally_predicable_of*.

c10 Behind the scenes, *izz* is set inclusion, so this is obvious.

c11-13 These contain modal operators which cannot be defined using this model.

c14-c15 I've not worked out what a "thing" is. Not even sure that I should have rendered Speranza's version using that term. However, I do think I know that *particular* and *universal* are opposites (contradictories), and hence I could conclude from these two conjectures that there can be no *forms* since they entail that a form is both individual and universal. Sounds like I have the wrong end of some stick or other.

c16 I don't know enough about "thing"s to prove this one.

c17 This is *c*01 in other words.

c18 This is *c*03*b* in other words.

c19 This is the contrapositive of the claim that *accidentally_predicable_of* is reflexive, which is false. Would be true for *essentially_predicable_of*, but we already have that stated directly as c17.

c20 Immediate from my definitions.

c21 This turns out to be false under my definitions, because I have not excluded the possibility of an empty predicate. However, one wonders why this should be excluded.

If I go over to a model adquate for modal operators then it will be easy to exclude this possibility.

c21 Fails for same reason as c20, though I could fix this by making the definition of *universal* insist on more than one member.

c23 I don't know why this should be true. Any particular which partakes of a form contradicts it.

c24 This is not true in the present model, because we might have only one particular, and hence no non-trivial forms.

c25-c26 These two tell me that forms are not substance, but attributes, which contradicts c24 which tells us that there are subtantial forms (if particulars are substances).

c27 I don't see why this should be true.

c28-c29 These two together entail that no universal is predicable of any particular.

c30 This says that nothing is essentially predicable of a particular except itself.

c31 This is a stronger version of c30 which says that, even if something isn't particular, nothing but itself is essentially predicable of it.

2.2 The Organon

The Organon is a collection of 6 books by Aristotle which form the main part of his work on Logic. The first of these is the Categories [**?**], on which Aristotle's Metaphyics depends. The Metaphysics, at least the parts involved in the Grice/Code analysis [**?**, **?**], is concerned with predication, which is also central to the formal core of Aristotle's logic, the theory of the syllogism, presented in the Prior Analytic, Book 1 [**?**].

Aristotle's account of syllogistic logic covers modal reasoning. In attempting to understand Aristotelian essentialism, one of the key problems is to establish the relationship between the two distinctions between necessary and contingent proposition, and between essential and accidental predication.

Though Code [**?**] does not conceive of himself as engaged in formalising Aristotle in a modern predicate logic, his presentation seems much closer to predicate logic than to syllogistic logic. I would like to understand the metaphysics if possible in terms of the kind of logic which Aristotle had at his disposal. To explore the extent to which this might be possible, some models of syllogistic logic might be helpful.

When we look at the syllogism with particular concern for the notion of predication involved, we find that the Grice's "izz/hazz" distinction (in Aristotle "said of" and "in") is not relevant. In this respect predication is simpler in the syllogism, but instead we have an orthogonal distinction into four kinds of predication according to whether the subject is universal or particular, and whether the predication is affirmative or negative, over which are later added the modal operators.

The semantics of the syllogism remains a matter of controversy in some respects. The majority of the syllogisms held to be valid by Aristotle would be valid if universal and particular propositions were translated as universal and existential quantification in a modern predicate logic. Four of the syllogisms held to be valid by Aristotle would not be sound under such an interpretation. These four are distinguished by having universal premises but a particular conclusion. I propose to call these the *universal-particular* syllogisms, which I may abbreviate *u-p*.

There are three most common approaches to the u-p syllogisms:

1. Consider them to be fallacious (possibly admitting the implicature).
2. Consider Aristotle's formal logic to be concerned exclusively with non-empty terms.
3. Consider universal propositions to be, in effect, the conjunction of a universal and an existential quantification.

However, it now seems that none of these are correct and that the approach closest to Aristotle's text may be to have existential import in assertions (whether universal or particular) and absent from denials.

2.2.1 Models and Their Significance

Three different methods of analysis are illustrated in a very simple way in application to the laws of direct inference and the assertoric (i.e. non-modal) syllogistic.

These consisting in interpreting or modelling *aspects* of Aristotelian logic using modern languages and tools. It is important to understand that the construction of an interpretation or a model says by itself nothing about Aristotelian logic until the author makes a claim about the relationship between the model and the thing modelled. It is never the intention that a model should be in all respects similar to the original.

In producing these models I prepare the model first, with only a vague sense of what it might be good for, and then make certain comparisons, and describe the results. Thus the first two interpretations, which may be called translations or reductions, serve only to reduce the question of truth or provability in Aristotelian logic to that of truth or provability in some other notation or deductive system.

This they do incompletely. The first interpretation fails to demonstrate a subset of the syllogisms (those which I here refer to as u-p syllogisms). Its purpose is therefore primarily to show that the most naive interpretations of Aristotle's language fail to validate his reasoning, and thereby to cast doubt on such interpretations.

The next set of models may also be understood in similar ways, but they can also be considered as attempts to capture the truth conditions of categorical assertions. The evaluation of these models is based again in the first instance on the correspondence between truth and provability between the model and the original. Only one of these models fails to show significant differences in these respects from Aristotle. It is therefore only in respect of the last of the models that we are motivated to enquire closer into the fidelity of the truth-conditional semantics which they exhibit.

The claims made for the models are very limited, and the purposes of the models are:

- to exhibit certain methods and consider their merits
- to provide a basis for formal reasoning using a modern proof tool in a manner consistent with Aristotle's logic
- to provide a basis for an exploration of Aristotle's metaphyics using the same tools and methods

Three methods of modelling are used. These methods may be described as:

- translation
- shallow embedding
- deep embedding

In a translation syntax is not preserved, the aristotelian assertions are translated into some other form altogether.

In an embedding of either depth, an attempt is made to preserve and work with the original syntax, though in practice there may be alterations to the syntax to enable the analysis to be undertaken using some preferred software for formal analysis. In this document the software is ProofPower.

A series of mappings into HOL have been used to evaluate these possibilities:

1. a set theoretic interpretation (Section 2.2.3)
2. an interpretation in propositional logic (Section 2.2.4)
3. a naive predicate calculus interpretation (Section 2.2.5)
4. predicate calculus with no empty terms (Section 2.2.6)
5. existential import in universals (Section 2.2.7)

6. existential import in affirmations (Section 2.2.8)
7. reasoning about deep embeddings
8. modal syllogisms (by non-emptyness) (Section 2.2.9)

Of these the first two are simplistic translations. The categorical propositions are mapped to sentences in HOL which resemble set theory, or the predicate calculus and by this means is obtained a partial reduction of the decision problem for truth of syllogisms.

The next three are 'shallow embeddings'. This means that the established representation of categorical assertions in the AEIO form is retained as syntax, so no translation of syntax is involved. These are given meaning by defining constants in HOL named 'a', 'e', 'i', and 'o' to capture the truth conditions of the four forms. This involves in the first instance a choice of type for the term variables and then the definition of these forms as functions from two such values to the type BOOL.

The interpretation is then evaluated by proving the soundness of various laws and syllogisms represented syntactically much as they are in the relevant literature.

Of these three shallow embeddings, the first two are based upon Strawson [**?**]. The third benefits from discussion on the *phil-logic* mailing list, and in particular on advice on the interpretation of Aristotle from Drake O'Brien supported by textual references which convinced me that neither of the Strawsonian interpretations could be correct.[2] These two points are firstly that Aristotle cannot be construed as presuming non-emptyness of terms (or as committing an existential fallacy), and that existential import is associated with the quality of the proposition, attaching to affirmative but not to negative propositions.

To facilitate comparison between the different interpretations the following list of 'features' will be checked. These are the various rules which might or might not be valid. My aim is to check which of these hold in each interpretation and then to summarise the results in a table.

- Laws of immediate inference.

  These fall into the following categories:

  - Simple conversion.
  - Conversions per accidens.
  - Obversion.
  - Contraposition and inversion.
  - The square of opposition.

- The syllogisms.

[2]This seems to be the accepted interpretation among contemporary scholars, and is described bt Terence Parsons in his "Overview of Aristotle's Logic" [**?**] available from his website.

The present treatment was based firstly on the wikipedia account of Aristotle's Logic, secondly on Strawson [**?**] from whom I first obtained the syllogisms not in Aristotle and began incorporation of direct inference, and then from Spade[**?**] I obtained the names for the extra syllogisms and a fuller account of how the syllogisms can be derived which is not yet reflected in the following. Using these sources I didn't get very close to the real Aristotle. Drake O'Brien, in discussion on *phil-logic* helped get be closer.

The main interest at present is the sequence of models which contributes to the formulation of an integrated model for the syllogism and the metaphysics in Section 2.3.

2.2.2 Preliminaries

Generating the Syllogisms

This following presentation of the forms of syllogism is based primarily on the one at wikipedia.

In the following several different interpretations of the syllogism are formalised as shallow semantic embeddings in HOL. Part of the evaluation of these interpretations consists in demonstrating that certain syllogisms are sound relative to the particular versions of the semantics. This is done by constructing a conjecture in HOL which will be true if the syllogism is sound, and then proving that the conjecture is true (i.e. proving the corresponding theorem). This procedure does not amount to a proof of the soundness of the syllogisms relative to the semantics, but gives almost the same level of confidence. A standard soundness proof would require a 'deep embedding', which would involve formalisation of the syntax of the syllogism and of the semantics as a relation between the syntax and the models. This would be considerably more arduous, and would not deliver comparably greater insights.

Before looking at alternative semantics for the syllogistic some machinery for generating syllogisms which will be used throughout is provided in this section.

Structurally a syllogism consists of two premises and a conclusion. The premises and conclusion are propositions, each of which is built from two terms in one of four ways. The different interpretations of the syllogism are obtained by chosing different types for the terms and different ways of constructing the propositions from the terms. For some purposes distinct ways of combining the propositions to form the completed syllogism will be necessary.

The information determining a particular interpretation will therefore be certain values in the meta-language:

mkAT a function which converts a name to a term of that name (having type: $_{\mathrm{ML}}\ulcorner string -> TERM \urcorner$)

mkAP a function which takes a propositional form and delivers a proposition constructor taking two terms and delivering a proposition (having type: $_{\mathrm{ML}}\ulcorner string -> (TERM * TERM) -> TERM \urcorner$)

mkAS a function which takes a list of propositions (the premises) and a single proposition (the conclusion) and delivers a syllogism as a sequent goal (having type: $_{\mathrm{ML}}\ulcorner (TERMlist * TERM) ->$

$GOAL\ulcorner)$

A choice about the semantics will be encapsulated as a record having these three components.

SML

```
type mapkit =
    {mkAT:string -> TERM,
     mkAP:string -> (TERM * TERM) -> TERM,
     mkAS:(TERM list * TERM) -> GOAL};
```

There are four forms of predication which are normally presented as infix operators over terms using the vowels “a”, “e”, “i”, “o”.

These are to be construed as follows:

HOL term	Meaning
⌜A a B⌝	All A are B
⌜A e B⌝	All A are not B
⌜A i B⌝	An A is B
⌜A o B⌝	An A is not B

Though exactly what these mean remains controversial and is a principle point of variation in the interpretations presented here.

Syllogisms come in four figures, according to the configuration of variables in the premises:

Figure 1 M-P, S-M ⊢ S-P
Figure 2 P-M, S-M ⊢ S-P
Figure 3 M-P, M-S ⊢ S-P
Figure 4 P-M, M-S ⊢ S-P

Where S, P and M are the subject, predicate and middle term respectively.

The following function generates a list of four quadruples of HOL variables which correspond to the four figures of syllogisms. It is parameterised by the HOL type used for predicates to that it can be re-used when we change the representation type.

SML

```
fun figureS mkt =
    let val M = mkt "M"
        and P = mkt "P"
        and S = mkt "𝕊"
    in [(M,P,S,M), (P,M,S,M), (M,P,M,S), (P,M,M,S)]
    end;
```

In most cases terms are mapped to variables of appropriate types for the chosen semantics. The following function when given a type will return a term constructor mapping term names to variables of that type.

SML
```
fun mkATvar tt s = mk_var(s,tt);
```

For these cases the list of figures with the relevant terms is obtained by this function:

SML
```
fun figurest tt = figureS (mkATvar tt);
```

and the following function when instantiated to some type will give the nth figure with terms as variables of that type.

SML
```
fun nthfig tt n = nth n (figurest tt);
```

These four figures are then repeated for each combination of the four types of premise in each of the premises and the conclusion. This gives $4 \times 4 \times 4 \times 4 = 256$ possibilities, of which 19 were held to be valid by Aristotle, four of them u-p syllogisms.

The use of vowels for the predicators allows the valid cases to be named using names in which the vowels tell you the form of the syllogism (if you also know the figure). The first vowel tells you the kind of syllogism in the first premise, the second vowel that in the second premise, and the third vowel that in the conclusion.

In the following table the names in square brackets are for u-p syllogisms. The names followed by exclamation marks are "subalternate mood", they do not appear in aristotle but are valid in the models here for which the u-p syllogisms hold [3].

Figure 1	Figure 2	Figure 3	Figure 4
Barbara	Cesare	[Darapti]	[Bramantip]
Celarent	Camestres	Disamis	Camenes
Darii	Festino	Datisi	Dimaris
Ferio	Baroco	[Felapton]	[Fesapo]
Barbari!	Cesaro!	Bocardo	Fresison
Celaront!	Camestrop!	Ferison	Camenop!

A syllogism may be identified either by the name in the above table (for the valid syllogisms), or by the three vowels, and the figure.

The above table is captured by the following definitions in our metalanguage:

[3] This I got from Spade [?].

We now define a data structure from which the valid syllogisms can be generated. With the model we are using only the 15 non-u-p syllogisms. They are shown in the following data structure.

SML

```
val syllogism_data1 =
    [("Barbara", 1),
    ("Celarent", 1),
    ("Darii", 1),
    ("Ferio", 1),
    ("Cesare", 2),
    ("Camestres", 2),
    ("Festino", 2),
    ("Baroco", 2),
    ("Disamis", 3),
    ("Datisi", 3),
    ("Bocardo", 3),
    ("Ferison", 3),
    ("Camenes", 4),
    ("Dimaris", 4),
    ("Fresison", 4)];
```

The rest of the table consists of "universal-particular" syllogisms, i.e. syllogisms with universal premises but a particular conclusion.

The following four u-p syllogisms were held to be valid by Aristotle (the ones in square brackets in the table).

SML

```
val syllogism_data2 =
    [("Darapti", 3),
    ("Felapton", 3),
    ("Bramantip", 4),
    ("Fesapo", 4)];
```

The following five further syllogisms are also sound, though not noted as such by Aristotle (the ones followed by exclamation marks in the table).

SML

```
val syllogism_data3 =
    [("Barbari", 1),
    ("Celaront", 1),
    ("Cesaro", 2),
```

```
        ("Camestrop", 2),
        ("Camenop", 4)];
```

We now assume available the triple of constructors mentioned above and define a function which converts syllogism identification in the form of a string of three vowels and a number in the range 1-4 into a syllogism. This is parameterised by a *mapkit* so that we can use it to deliver several different interpretations of the syllogism.

SML

```
fun vowels_from_string s = filter (fn x => x mem (explode "aeiou")) (explode s);

fun mkSyll (sk:mapkit) (s, n) =
        let val [pa, pb, co] = vowels_from_string s
            val (a,b,c,d) = nth (n-1) (figureS (#mkAT sk))
            val pl = [#mkAP sk pa (a,b), #mkAP sk pb (c,d)]
            val c = #mkAP sk co (#mkAT sk "𝕊", #mkAT sk "P")
        in #mkAS sk (pl, c)
        end;
```

We can then map this operation over the a list of syllogism data as follows:

SML

```
fun mkGoals sk = map (fn (s,n) => (s, mkSyll sk (s, n)));
```

If we further supply a proof tactic and a labelling suffix then we can prove a whole list of goals and store the results in the current theory.

SML

```
fun proveGoals tac suff = map
        (fn (s,g) =>
              (s,
               save_thm (s^suff, tac_proof (g, tac))
                    handle _ => t_thm));
```

SML

```
fun proveSylls sk tac suff sd = proveGoals tac suff (mkGoals sk sd);
```

The following functions take a string which is the name of a syllogism extract the vowels which occur in it and convert them into the corresponding predication operator to give a triple of operators.

SML
```
fun op_from_char ot c = mk_const (if c = "o" then "u" else c, ot);

fun optrip_from_text ot s =
    let val [a, b, c] = (map (op_from_char ot) o vowels_from_string) s;
    in (a, b, c)
    end;
```

A Common Special Case The most common pattern is that:

- Terms are mapped to variables of some type.
- Propositions are formed using infix operators *a, e, i, o*.
- Syllogisms are sequents, the premises in the assumptions.

The following constructs a mapkit for such cases when supplied with the type of the term variables, and the four proposition constructors.

SML
```
fun mkSimpMapkit ty opl =
    let val mkcop = fn s => snd (find (combine (explode "aeio") opl) (fn (n,v) => n = s))
    in
        {mkAT = fn s => mk_var(s, ty),
        mkAP = fn s => fn (l,r) => mk_app(mk_app (mkcop s, l), r),
        mkAS = fn x => x}
    end;
```

The Square of Opposition

A	contraries	E
subalternates	contradictories	subalternates
I	subcontraries	O

Table 2.1: The square of Opposition 1

The relationships between corners of the square of opposition have the following names:

The full set of relationships exhibited by the square is:

The goal conclusions for theorems expressing compliance are (details of syntax will vary between interpretations):

```
SML
val ao_contrad = ⌜A a B ⇔ ¬ A o B⌝;
val ei_contrad = ⌜A e B ⇔ ¬ A i B⌝;
val ae_contrar = ⌜¬ (A a B ∧ A e B)⌝;
val io_subcont = ⌜A i B ∨ A o B⌝;
val ai_subalt = ⌜A a B ⇒ A i B⌝;
val eo_subalt = ⌜A e B ⇒ A o B⌝;
```

This provides a compact way of comparing different interpretations of the assertoric forms.

The principal point of difference is in subalternation which fails when the quantifiers are interpreted in the modern sense and empty terms are admitted. One remedy for this defect might be to add existential import to the universal quantifier, but this by itself would cause the contradictories to fail. Another remedy is to add existential import to the universal and define the particular modes through the contradictories.

Peter Strawson enumerates 14 laws of immediate inference, not all of which seem to have been embraced by Aristotle. The list nevertheless provides a good collection against which an interpretation of the propositions can be evaluated, along with the 24 syllogisms.

The following table shows the correspondence between the numbers used by Strawson, the names used here, and a fuller name.

Are The Syllogisms Tautologous?

This section is a kind of preliminary skirmish provoked by some discussions on phil-logic in which one or two specific questions are addressed.

The question which provoked this section was:

- Are the valid syllogisms tautologous?

This is complex because it depends on a view about the semantics of the syllogism and also upon what particular notion of tautology is in play.

contradictories	A and B are contradictories if $\ulcorner A \Leftrightarrow \neg B \urcorner$
contraries	A and B are contraries if they are not both true.
subcontraries	A and B are sub-contraries if they are not both false.
subalternates	B is subalternate to A if A implies B

Table 2.2: The square of Opposition 2

Two related questions come from remarks by P.V.Spade. P.V. Spade, in a thumbnail history of logic [**?**] remarks that some later peripatetics attempted to show that stoic propositional logic was simply the syllogism in other clothes, and others that the two are in some sense equivalent.

- Is there any sense in which the syllogism can be said to encompass Stoic propositional logic?
- Is there any sense in which the syllogism might be seen to be equivalent to Stoic propositional logic?

I have no knowledge of Stoic propositional logic, and this section is therefore concerned with the relationship between the syllogism and modern propositional logic, which I presume is a superset (probably but not necessarily proper) of Stoic propositional logic.

I distinguish three notions of tautology (though this is not intended to be an exhaustive list) on which something can be said in this connection.

propositional A valid sentence of the propositional calculus.

first order A valid sentence of first order logic.

analytic A sentence whose truth conditions assign truth to it in every situation or interpretation consistent with the meanings of the language in which it is expressed.

The connection between these is in the notions of truth function and truth condition. The first two are notions of truth functional tautology. In both languages the truth value of a sentence is a function of the truth values of the atomic sentences it contains, and a sentence is tautologous if that truth function is the one which always delivers the value "true". In the second of these two the notion of truth function is slightly more complex and some explanation may be necessary of how the quantifiers can be so construed. It may be noted however that it is a thesis of Wittgenstein's Tractatus[**?**] that logical truth in general is tautologous. In the last case the restriction to truth functions (i.e. functions which take truth values as arguments and deliver truth values as results) is dropped, allowing that the essential element is that the relevant part of the semantics can be rendered as *truth conditions* in which the result is a truth value but the argument need not be, and that it is of the essence of the concept

ao_contrad	contradictories
ei_contrad	contradictories
ae_contrar	contraries
io_subcont	subcontraries
ai_subalt	subalternates
eo_subalt	subalternates

Table 2.3: The square of Opposition 3

tautology to single out from truth functions or conditions in general those which always deliver the truth value 'true'.

In relation to these three notions of tautology, one thinks naturally of categorical propositions as involving quantification, and therefore, of syllogisms, if tautologous being so at best as first order validities, but in any case, if sound, as analytic. These intuitions do not support a connection between syllogisms and propositional tautologies.

The connection we illustrate here is obtained via elementary set theory. The intuition behind this is that Aristotelian predication is analogous to set inclusion, taking terms to denote their extensions (and singular terms as denoting singleton sets), together with complementation and negation. This works except for the problematic univeral-particular syllogisms which infer particular conclusions from universal premises. A connection with propositional logic can be obtained from this, since the elementary set theory required for this interpretation is a boolean algebra, as is the propositional calculus.

Partial reductions to set theory and propositional logic are exhibited in sections 2.2.3 and 2.2.4. The reduction to set theory is semantically more plausible than the second step to proposional logic, and provides an account of those u-p syllogisms, which are not accounted for by the propositional reduction.

We obtain kind of reduction of the decision problem for the non-u-p syllogisms to the decision problem for truth functional tautologies, but only in a very informal sense, since there are only a small finite set of valid syllogisms (though schemata) and so the decision problem is trivial. For this reason syllogistic logic cannot be as expressive as a modern propositional logic, but conceivably might be closer to Stoic propositional logic.

The first step, to set theory is illustrated in the Wikpedia article on the syllogism, which gives Venn diagrams for all 24 "valid" syllogisms.

Though it is convenient to think of this as a two stage reduction, in the following implementation the two different reductions are generated independently.

2.2.3 Interpretation in Set Theory

SML

```
open_theory "aristotle";
force_new_theory "syllog1";
```

Under the proposed mapping terms of the syllogism are mapped to propositions. The term "Mortal" may be thought of as being mapped to the proposition "x is Mortal".

Thus the Barbara syllogism:

- All As are Bs
- All Bs are Cs
- =========

- All As are Cs

might be rendered set theoretically as:

$(A \subseteq B \wedge B \subseteq C) \Rightarrow (A \subseteq C)$

where A, B and C are subsets of some universal set.

However, since we are working here in a sequent calculus we can render the theorems closely to the original as:

$$\begin{array}{l} [A \subseteq B, \\ \quad B \subseteq C] \\ \vdash A \subseteq C \end{array}$$

A set theoretic presentation of the non-u-p syllogisms can be obtained by assuming that the sets are not empty, but no similarly plausible account of these syllogisms is available in propositional logic.

Here is a table describing the proposed mappings:

Aristotle	Meaning	Set Theory	Propositional Analogue
A a B	All A are B	A $\subseteq$ B	A $\Rightarrow$ B
A e B	All A are not B	$A \subseteq \sim B$	A $\Rightarrow \neg$B
A i B	An A is B	$\neg(A \subseteq \sim B)$	$\neg$(A $\Rightarrow \neg$B)
A o B	An A is not B	$\neg$(A $\subseteq$ B)	$\neg$(A $\Rightarrow$ B)

These mappings can be encapsulated as two 'mapkits'.

First the set theory mapping:

SML

```
val st_mkAP =
    let   fun a (su, pr) = ⌜MLsu⌝ ⊆ ⌜MLpr⌝⌝
          fun e (su, pr) = ⌜MLsu⌝ ⊆ ∼ ⌜MLpr⌝⌝
          fun i (su, pr) = ⌜¬ MLsu⌝ ⊆ ∼ ⌜MLpr⌝⌝
          fun u (su, pr) = ⌜¬ MLsu⌝ ⊆ ⌜MLpr⌝⌝
    in fn s => case s of "a" => a | "e" => e | "i" => i | "o" => u
    end;

declare_type_abbrev("TermS", [], ⌜:'a SET⌝);

val st_mapkit:mapkit =
    {     mkAT = fn s => mk_var(s, ⌜:TermS⌝),
          mkAP = st_mkAP,
          mkAS = fn x => x};
```

Generating The Propositions

This section is mainly given over to short programs in our metalanguage the end effect of which is to secure the proof of the 15 theorems of set theory and 15 propositional tautologies which are obtained from non-u-p syllogisms by this naive transformation.

The results are visible in the "theorems" section of the theory listing in listing of theory *syllog1*[**?**], and this section can be safely skipped by anyone whose interest is purely philosophical.

Mostly this uses functionality devised in the previous section to achieve two slightly different sets of theorems.

The following functions take a string which is the name of a syllogism, extract the vowels which occur in it and convert them into the corresponding propositional formula to give a triple of propositions.

giving the following list of "goals":

```
val it =
  [ ([⌜M ⊆ P⌝, ⌜𝕊 ⊆ M⌝], ⌜𝕊 ⊆ P⌝),
    ([⌜M ⊆ ∼ P⌝, ⌜𝕊 ⊆ M⌝], ⌜𝕊 ⊆ ∼ P⌝),
    ([⌜M ⊆ P⌝, ⌜¬ 𝕊 ⊆ ∼ M⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜M ⊆ ∼ P⌝, ⌜¬ 𝕊 ⊆ ∼ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜P ⊆ ∼ M⌝, ⌜𝕊 ⊆ M⌝], ⌜𝕊 ⊆ ∼ P⌝),
    ([⌜P ⊆ M⌝, ⌜𝕊 ⊆ ∼ M⌝], ⌜𝕊 ⊆ ∼ P⌝),
    ([⌜P ⊆ ∼ M⌝, ⌜¬ 𝕊 ⊆ ∼ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜P ⊆ M⌝, ⌜¬ 𝕊 ⊆ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ M ⊆ ∼ P⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜M ⊆ P⌝, ⌜¬ M ⊆ ∼ 𝕊⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜¬ M ⊆ P⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜M ⊆ ∼ P⌝, ⌜¬ M ⊆ ∼ 𝕊⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜P ⊆ M⌝, ⌜M ⊆ ∼ 𝕊⌝], ⌜𝕊 ⊆ ∼ P⌝),
    ([⌜¬ P ⊆ ∼ M⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
```

Proving the Syllogisms

Here is a proof tactic for the non-u-p syllogisms under the set theory interpretation.

SML

```
val st_tac = REPEAT (POP_ASM_T ante_tac)
             THEN PC_T1 "hol1" rewrite_tac []
             THEN prove_tac[];
```

We now apply this to the non-u-p syllogisms, saving the results in the theory (listing of theory *syllog1*[**?**]).

SML

```
proveSylls st_mapkit st_tac "" syllogism_data1;
```

We now adjust the set theoretic reduction to deliver u-p syllogisms by including a non-emptyness assumption and obtain proofs of the nine syllogisms valid u-p syllogisms.

giving the following list of "goals":

```
val it =
  [ ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜M ⊆ P⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜M ⊆ ∼ P⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜P ⊆ M⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜P ⊆ ∼ M⌝, ⌜M ⊆ 𝕊⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜M ⊆ P⌝, ⌜𝕊 ⊆ M⌝], ⌜¬ 𝕊 ⊆ ∼ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜M ⊆ ∼ P⌝, ⌜𝕊 ⊆ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜P ⊆ ∼ M⌝, ⌜𝕊 ⊆ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜P ⊆ M⌝, ⌜𝕊 ⊆ ∼ M⌝], ⌜¬ 𝕊 ⊆ P⌝),
    ([⌜¬ {} ∈ {M; 𝕊; P}⌝, ⌜P ⊆ M⌝, ⌜M ⊆ ∼ 𝕊⌝], ⌜¬ 𝕊 ⊆ P⌝)]
: (TERM list ∗ TERM) list
```

We now apply this to the non-u-p syllogisms, saving the results in the theory.

The following code generates the goals for proving the above syllogisms from the previously defined data structure desribing the valid syllogisms, generates and checks formal proofs and saves the resulting theorems in the current theory.

SML

```
val st2_mapkit:mapkit =
      {     mkAT = fn s => mk_var(s, ⌜:TermS⌝),
            mkAP = st_mkAP,
            mkAS = fn (ps, c) => (⌜¬ ({}:'a SET) ∈ {M; 𝕊; P}⌝::ps, c)};
```

SML

```
val st2_tac = REPEAT (POP_ASM_T ante_tac)
            THEN PC_T1 "hol1" rewrite_tac [] THEN prove_tac[];
```

SML

```
proveSylls st2_mapkit st2_tac "" (syllogism_data2 @ syllogism_data3);
```

The full set of syllogisms may be found in the theory (listing of theory *syllog1*[**?**]).

2.2.4 Propositional Interpretation

SML

```
open_theory "aristotle";
force_new_theory "syllog2";
```

The term “Mortal” may be thought of as being mapped to the proposition “x is Mortal”.

Thus the Barbara syllogism:

- All As are Bs
- All Bs are Cs
- ==========
- All As are Cs

is rendered in the propositional calculus as:

$(A \Rightarrow B \wedge B \Rightarrow C) \Rightarrow (A \Rightarrow C)$

where A, B and C are boolean values or propositions.

However, since we are working here in a sequent calculus we can render the theorems closely to the original as:

$$[A \Rightarrow B,\ B \Rightarrow C] \vdash A \Rightarrow C$$

The propositional logic mapping is determined as follows:

SML

```
declare_type_abbrev("TermP", [], ⌜:BOOL⌝);

val pl_mapkit:mapkit =
      let fun a (su, pr) = ⌜ML⌞su⌟ ⇒ ML⌞pr⌟⌝
          fun e (su, pr) = ⌜ML⌞su⌟ ⇒ ¬ ML⌞pr⌟⌝
          fun i (su, pr) = ⌜¬ (ML⌞su⌟ ⇒ ¬ ML⌞pr⌟)⌝
          fun u (su, pr) = ⌜¬( ML⌞su⌟ ⇒ ML⌞pr⌟)⌝
      in { mkAT = fn s => mk_var(s, ⌜:TermP⌝),
          mkAP = fn s => case s of
                "a" => a
```

```
            |     "e" => e
            |     "i" => i
            |     "o" => u,
            mkAS = fn x => x}
      end;
```

giving the following list of "goals":

```
val it =
  [([⌜M ⇒ P⌝, ⌜𝕊 ⇒ M⌝], ⌜𝕊 ⇒ P⌝),
    ([⌜M ⇒ ¬ P⌝, ⌜𝕊 ⇒ M⌝], ⌜𝕊 ⇒ ¬ P⌝),
    ([⌜M ⇒ P⌝, ⌜¬ (𝕊 ⇒ ¬ M)⌝], ⌜¬ (𝕊 ⇒ ¬ P)⌝),
    ([⌜M ⇒ ¬ P⌝, ⌜¬ (𝕊 ⇒ ¬ M)⌝], ⌜¬ (𝕊 ⇒ P)⌝),
    ([⌜P ⇒ ¬ M⌝, ⌜𝕊 ⇒ M⌝], ⌜𝕊 ⇒ ¬ P⌝),
    ([⌜P ⇒ M⌝, ⌜𝕊 ⇒ ¬ M⌝], ⌜𝕊 ⇒ ¬ P⌝),
    ([⌜P ⇒ ¬ M⌝, ⌜¬ (𝕊 ⇒ ¬ M)⌝], ⌜¬ (𝕊 ⇒ P)⌝),
    ([⌜P ⇒ M⌝, ⌜¬ (𝕊 ⇒ M)⌝], ⌜¬ (𝕊 ⇒ P)⌝),
    ([⌜¬ (M ⇒ ¬ P)⌝, ⌜M ⇒ 𝕊⌝], ⌜¬ (𝕊 ⇒ ¬ P)⌝),
    ([⌜M ⇒ P⌝, ⌜¬ (M ⇒ ¬ 𝕊)⌝], ⌜¬ (𝕊 ⇒ ¬ P)⌝),
    ([⌜¬ (M ⇒ P)⌝, ⌜M ⇒ 𝕊⌝], ⌜¬ (𝕊 ⇒ P)⌝),
    ([⌜M ⇒ ¬ P⌝, ⌜¬ (M ⇒ ¬ 𝕊)⌝], ⌜¬ (𝕊 ⇒ P)⌝),
    ([⌜P ⇒ M⌝, ⌜M ⇒ ¬ 𝕊⌝], ⌜𝕊 ⇒ ¬ P⌝),
    ([⌜¬ (P ⇒ ¬ M)⌝, ⌜M ⇒ 𝕊⌝], ⌜¬ (𝕊 ⇒ ¬ P)⌝),
    ([⌜P ⇒ ¬ M⌝, ⌜¬ (M ⇒ ¬ 𝕊)⌝], ⌜¬ (𝕊 ⇒ P)⌝)] : (TERM list * TERM) list
```

The following tactic suffices for proving propositional tautologies.

SML

```
val pl_tac = REPEAT (POP_ASM_T ante_tac)
             THEN REPEAT strip_tac;
```

We now apply this to the non-u-p syllogisms, saving the results in the theory.

SML

```
val valid_psylls = proveSylls pl_mapkit pl_tac "" syllogism_data1;
```

This is the resulting value.

```
val valid_psylls =
  [("Barbara", M ⇒ P, 𝕊 ⇒ M ⊢ 𝕊 ⇒ P),
    ("Celarent", M ⇒ ¬ P, 𝕊 ⇒ M ⊢ 𝕊 ⇒ ¬ P),
    ("Darii", M ⇒ P, ¬ (𝕊 ⇒ ¬ M) ⊢ ¬ (𝕊 ⇒ ¬ P)),
    ("Ferio", M ⇒ ¬ P, ¬ (𝕊 ⇒ ¬ M) ⊢ ¬ (𝕊 ⇒ P)),
    ("Cesare", P ⇒ ¬ M, 𝕊 ⇒ M ⊢ 𝕊 ⇒ ¬ P),
    ("Camestres", P ⇒ M, 𝕊 ⇒ ¬ M ⊢ 𝕊 ⇒ ¬ P),
    ("Festino", P ⇒ ¬ M, ¬ (𝕊 ⇒ ¬ M) ⊢ ¬ (𝕊 ⇒ P)),
    ("Baroco", P ⇒ M, ¬ (𝕊 ⇒ M) ⊢ ¬ (𝕊 ⇒ P)),
    ("Disamis", ¬ (M ⇒ ¬ P), M ⇒ 𝕊 ⊢ ¬ (𝕊 ⇒ ¬ P)),
    ("Datisi", M ⇒ P, ¬ (M ⇒ ¬ 𝕊) ⊢ ¬ (𝕊 ⇒ ¬ P)),
    ("Bocardo", ¬ (M ⇒ P), M ⇒ 𝕊 ⊢ ¬ (𝕊 ⇒ P)),
    ("Ferison", M ⇒ ¬ P, ¬ (M ⇒ ¬ 𝕊) ⊢ ¬ (𝕊 ⇒ P)),
    ("Camenes", P ⇒ M, M ⇒ ¬ 𝕊 ⊢ 𝕊 ⇒ ¬ P),
    ("Dimaris", ¬ (P ⇒ ¬ M), M ⇒ 𝕊 ⊢ ¬ (𝕊 ⇒ ¬ P)),
    ("Fresison", P ⇒ ¬ M, ¬ (M ⇒ ¬ 𝕊) ⊢ ¬ (𝕊 ⇒ P))] : (string * THM) list
```

.. which is a list of name/theorem pairs of the tautologies corresponding to each syllogism.

The theorems are also displayed in the theory listing in listing of theory *syllog2*[**?**]

Some words about the very limited signficance of this little exercise would be appropriate here!

2.2.5 Naive Interpretation in Predicate Calculus

SML

```
open_theory "aristotle";
force_new_theory "syllog3";
```

Semantics

Aristotle's syllogistic logic is concerned with inferences between judgements considered as predications. A predication in Aristotle affirms a *predicate* of some *subject*, but by contrast with more recent notions of predication the subject need not be an individual, the kinds of things which appear as predicates may also appear as subjects, and the relationship expressed seems closer to a modern eye to set inclusion than to what we now regard as predication. Since subject and predicate are for present purposes the same kind of thing, it is useful to have a name for that kind of thing, and I will use the name *property*.

There are four kinds of predication which we have here to account for, which we will do by offering definitions which provide a good model for syllogistic logic, i.e. one in which the syllogisms held to be true by Aristotle are in fact true. Before providing these definitions we must decide what kind of thing are the terms which are related by Aristotelian predication.

In HOL the most natural answer to this is "boolean valued functions" which are objects of type $\ulcorner : 'a \to BOOL \urcorner$ for some type of individuals which we can leave open by using the type variable $\ulcorner : 'a \urcorner$. This provides a simple model of Aristotle's non-u-p syllogistic reasoning. Four of the syllogisms which Aristotle considered valid fail under this conception of predicate, because among the objects of type $\ulcorner : 'a \to BOOL \urcorner$ is the function $\ulcorner \lambda x :' a \bullet F \urcorner$ which corresponds to a predicate with empty extension and does not admit inference from the universal to the existential (unless the universal is interpreted specially).

SML

```
declare_type_abbrev("Term2", [], ⌜:'a → BOOL⌝);
```

Predication

"o" is already in use for functional composition, so we will use "u" instead and then use an alias to permit us to write this as "o" (type inference will usually resolve any ambiguity).

The predication operators are defined as follows:

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : Term2 \to Term2 \to BOOL$$

$$\forall A\ B \bullet\ A\ a\ B \Leftrightarrow \forall x \bullet\ A\ x \Rightarrow B\ x$$

HOL Constant

$$\$\boldsymbol{e} : Term2 \to Term2 \to BOOL$$

$$\forall A\ B \bullet\ A\ e\ B \Leftrightarrow \forall x \bullet\ A\ x \Rightarrow \neg\ B\ x$$

HOL Constant

$$\$\boldsymbol{i} : Term2 \to Term2 \to BOOL$$

$$\forall A\ B \bullet\ A\ i\ B \Leftrightarrow \exists x \bullet\ A\ x \wedge B\ x$$

HOL Constant

$$\$\boldsymbol{u} : \mathit{Term2} \rightarrow \mathit{Term2} \rightarrow \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ u\ B \Leftrightarrow \exists x \bullet\ A\ x \wedge \neg\ B\ x$$

SML

```
declare_alias("o", ⌜$u⌝);
```

Note that as defined above these come in complementary pairs, a being the negation of o and e of i. If we had negation we could manage with just two predication operators.

The Laws of Immediate Inference

Though in the source of this kind of "literate script" are to be found the scripts for generating and checking the proofs of all the theorems presente, it will not be my practice to expose these scripts in the printed version of the document. These scripts are not usually intelligible other than in that intimate man-machine dialogue which they mediate, and sufficient knowledge for most purposes of the structure of the proof will be found in the detailed lemmas proven (since the level of proof automation is modest).

However, I will begin by exposing some of the scripts used for obtaining proofs of syllogisms in this model, to give the reader an impession of the level of complexity and kind of obscurity involved in this kind of formal work, I will not attempt sufficient explanation to make these scripts intelligible, they are best understood in the interactive environment, all the scripts are available for readers who want to run them.

Most readers are expected to skip over the gory details, the philosophical points at stake do not depend on the details of the proofs.

Before addressing the laws of immediate inference [4] I devise a tactic for automating simple proofs in this domain.

The following elementary tactic expands the goal by applying the definitions of the operators and then invokes a general tactic for the predicate calculus. A rule is also defined using that tactic for direct rather than interactive proof.

SML

```
val syll_tac = asm_prove_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝]);
fun syll_rule g = tac_proof (g, syll_tac);
```

[4] in which I followed Strawson [**?**], though I can now cite Aristotle, Prior Analytic, Book 1, Part 2. [**?**]

Simple Conversion Using the above tactic thus:

SML

```
val e_conv_thm = save_thm ("e_conv_thm", syll_rule([⌜A e B⌝], ⌜B e A⌝));
val i_conv_thm = save_thm ("i_conv_thm", syll_rule([⌜A i B⌝], ⌜B i A⌝));
```

The following two theorems are obtained.

```
val e_conv_thm = A e B ⊢ B e A : THM
val i_conv_thm = A i B ⊢ B i A : THM
```

The following two theorems show that the other obvious conversions are false.

```
a_not_conv_thm = ⊢ ∃ A B• A a B ∧ ¬ B a A
o_not_conv_thm = ⊢ ∃ A B• A o B ∧ ¬ B o A
```

Conversion Per Accidens These don't work here because they rely upon the u-p syllogisms.

Obversion For these we need to define an operation of complementation on terms.

HOL Constant

```
Complement : Term2 → Term2
```

```
∀A α• (Complement A) α ⇔ ¬ (A α)
```

We will use "~" as a shorthand for "complement".

SML

```
declare_alias ("∼", ⌜Complement⌝);
```

```
ae_obv_thm =   A a B ⊢ A e ∼ B
ea_obv_thm =   A e B ⊢ A a ∼ B
io_obv_thm =   A i B ⊢ A o ∼ B
oi_obv_thm =   A o B ⊢ A i ∼ B
```

Contraposition and Inversion

The Square of Opposition

```
ao_contrad_thm = ⊢ A a B ⇔ ¬ A o B
ei_contrad_thm = ⊢ A e B ⇔ ¬ A i B
¬ae_contrar_thm = ⊢ ¬ (∀ A B• ¬ (A a B ∧ A e B))
¬io_subcont_thm = ⊢ ¬ (∀ A B• A i B ∨ A o B)
¬ai_subalt_thm = ⊢ ¬ (∀ A B• A a B ⇒ A i B)
¬eo_subalt_thm = ⊢ ¬ (∀ A B• A e B ⇒ A o B)
```

The Syllogisms

The fifteen valid non-u-p syllogisms are true under this semantics and can be proven formally with ease.

Generating Syllogisms

First we make a *mapkit*.

SML

```
val s1mapkit:mapkit = mkSimpMapkit ⌜:Term2⌝ [⌜$a⌝,⌜$e⌝,⌜$i⌝,⌜$u⌝];
```

Then we apply this in generating and proving the non u-p syllogisms.

SML

```
proveGoals syll_tac "" (mkGoals s1mapkit syllogism_data1);
```

This is the resulting value.

```
val valid_sylls = [
      ("Barbara", M a P, 𝕊 a M ⊢ 𝕊 a P),
      ("Celarent", M e P, 𝕊 a M ⊢ 𝕊 e P),
      ("Darii", M a P, 𝕊 i M ⊢ 𝕊 i P),
      ("Ferio", M e P, 𝕊 i M ⊢ 𝕊 o P),
      ("Cesare", P e M, 𝕊 a M ⊢ 𝕊 e P),
      ("Camestres", P a M, 𝕊 e M ⊢ 𝕊 e P),
      ("Festino", P e M, 𝕊 i M ⊢ 𝕊 o P),
      ("Baroco", P a M, 𝕊 o M ⊢ 𝕊 o P),
      ("Disamis", M i P, M a 𝕊 ⊢ 𝕊 i P),
      ("Datisi", M a P, M i 𝕊 ⊢ 𝕊 i P),
      ("Bocardo", M o P, M a 𝕊 ⊢ 𝕊 o P),
      ("Ferison", M e P, M i 𝕊 ⊢ 𝕊 o P),
```

```
        ("Camenes", P a M, M e S ⊢ S e P),
        ("Dimaris", P i M, M a S ⊢ S i P),
        ("Fresison", P e M, M i S ⊢ S o P)
] : (string * THM) list
```

The theorems are also displayed in the theory listing in listing of theory *syllog3*[**?**]

2.2.6 Predicate Calculus Without Empty Terms

There is more than one way in which the semantics of the syllogism can be modified to make the inference from "All As are Bs" to "Some As are Bs" sound. One way would be to change the meaning of "All". This would interfere with the square of opposition by making diagonal entries no longer contradictories. From this I initially inferred that the exclusion of empty predicates is a better approach. [5]

We can then prove valid 24 forms of syllogism.

For the most concise statement of the results of the exercise, the reader should refer directly to the theory listing in listing of theory *syllog4*[**?**].

SML

```
open_theory "aristotle";
force_new_theory "syllog4";
```

Semantics

The key to getting the u-p syllogisms into the model is the adoption of a type for the variables in the syllogisms which does not include empty predicates. We could do this by defining a new type which is a sub-type of the propositional functions, but it is simpler to use another type-abbreviation as follows.

Instead of using a propositional function, which might be unsatisfiable, we use an ordered pair. The pair consists of one value, a value for which the predicate is true, and a propositional function. The predicate (v, pf) is then to be considered true of some value x *either* if x is v *or* if pf is true of x.

SML

```
declare_type_abbrev("Term3", [], ⌜:'a × ('a → BOOL)⌝);
```

Predication

To work with this new type for the predicates we define a function which will convert this kind of predicate into the old kind, as follows:

[5] This seemed then to be endorsed by Robin Smith in the Stanford Encyclopaedia of Philosophy [**?**].

HOL Constant

$$\boldsymbol{p} : \mathit{Term3} \to (\mathit{'a} \to \mathit{BOOL})$$

$$\forall A\bullet\ p\ A = \lambda x\bullet\ \mathit{let}\ (v,f) = A\ \mathit{in}\ x = v \vee f\ x$$

The resulting values have the same type as the old, but they will never have empty extension.

The following principle can be proven (proof omitted):

$$p\_\exists\_lemma =$$
$$\vdash \forall\ A\bullet\ \exists v\bullet\ p\ A\ v$$

This principle is what we need to prove the u-p syllogisms.

It should be noted that there is no complementation operation on terms of this type and that the obversions will therefore fail.

We then proceed in a similar manner to the first model, using the function p to convert the new kind of predicate into the old.

They predication operators are then defined. Note that the differences are small and uniform. The type $\ulcorner : \mathit{PROP} \urcorner$ is changed to $\ulcorner : \mathit{Term3} \urcorner$ and the function p is invoked before applying a predicate.

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : \mathit{Term3} \to \mathit{Term3} \to \mathit{BOOL}$$

$$\forall A\ B\bullet\ A\ a\ B \Leftrightarrow \forall x\bullet\ p\ A\ x \Rightarrow p\ B\ x$$

HOL Constant

$$\$\boldsymbol{e} : \mathit{Term3} \to \mathit{Term3} \to \mathit{BOOL}$$

$$\forall A\ B\bullet\ A\ e\ B \Leftrightarrow \forall x\bullet\ p\ A\ x \Rightarrow \neg\ p\ B\ x$$

HOL Constant

$$\$\boldsymbol{i} : \mathit{Term3} \to \mathit{Term3} \to \mathit{BOOL}$$

$$\forall A\ B\bullet\ A\ i\ B \Leftrightarrow \exists x\bullet\ p\ A\ x \wedge p\ B\ x$$

HOL Constant

$$\$\boldsymbol{u} : \mathit{Term3} \to \mathit{Term3} \to \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ u\ B \Leftrightarrow \exists x \bullet\ p\ A\ x \wedge \neg\ p\ B\ x$$

SML

```
declare_alias("o", ⌜$u⌝);
```

With these defined we can now produce a 'mapkit' for translating the syllogisms under this semantics.

SML

```
val s2_mapkit:mapkit = mkSimpMapkit ⌜:Term3⌝ [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝];
```

Laws of Immediate Inference

The same tactic used for proof of the syllogisms in the previous model still works with this model (with the new definitions), but does not prove the u-p syllogisms.

To obtain proofs of these other syllogisms we need to make use of the lemma we proved about p, *p_∃_lemma*. This we do by instantiating it for each of the variables which appear in the syllogisms and supplying these for use in the proof.

SML

```
val syll_tac2 =
      (MAP_EVERY (fn x => strip_asm_tac (∀_elim x p_∃_lemma))
            [⌜M:Term3⌝, ⌜P:Term3⌝, ⌜𝕊:Term3⌝, ⌜A:Term3⌝, ⌜B:Term3⌝])
       THEN asm_prove_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝]);

fun syll_rule2 g = tac_proof(g, syll_tac2);
```

SML

```
val e_conv = ([⌜A e B⌝], ⌜B e A⌝);
val i_conv = ([⌜A i B⌝], ⌜B i A⌝);
```

SML

```
val e_conv_thm = save_thm ("e_conv_thm", syll_rule2 e_conv);
val i_conv_thm = save_thm ("i_conv_thm", syll_rule2 i_conv);
```

```
val e_conv_thm = A e B ⊢ B e A : THM
val i_conv_thm = A i B ⊢ B i A : THM
```

Simple Conversion In this version of the semantics, "a" and "o" conversion is neither provable nor refutable. In the previous version, since the universe is a HOL type there is at least one individual, and contradictory predicates are allowed, we can use these two to disprove the two conversions. With this semantics there is no empty predicate, and we cannot know that there are two distinct predicates.

SML
```
val sg_03 = ([⌜A a B⌝], ⌜B i A⌝);
val sg_04 = ([⌜A e B⌝], ⌜B u A⌝);
```

SML
```
val ai_conv_thm = save_thm ("ai_conv_thm", syll_rule2 sg_03);
val eo_conv_thm = save_thm ("eo_conv_thm", syll_rule2 sg_04);
```

```
val ai_conv_thm = A a B ⊢ B i A : THM
val eo_conv_thm = A e B ⊢ B o A : THM
```

Conversion Per Accidens

Obversion We have been unable to define a complementation operation and the obversions listed by Strawson cannot even be expressed in this representation.

SML
```
val sg_09 = ([]:TERM list, ⌜A a B ⇔ ¬ A u B⌝);
val sg_10 = ([]:TERM list, ⌜A e B ⇔ ¬ A i B⌝);
val sg_11 = ([]:TERM list, ⌜¬ (A a B ∧ A e B)⌝);
val sg_12 = ([]:TERM list, ⌜A i B ∨ A u B⌝);
val sg_13 = ([⌜A a B⌝], ⌜A i B⌝);
val sg_14 = ([⌜A e B⌝], ⌜A u B⌝);
```

SML
```
val sg_09_thm = save_thm ("sg_09_thm", syll_rule2 sg_09);
val sg_10_thm = save_thm ("sg_10_thm", syll_rule2 sg_10);
val sg_11_thm = save_thm ("sg_11_thm", syll_rule2 sg_11);
val sg_12_thm = save_thm ("sg_12_thm", syll_rule2 sg_12);
val sg_13_thm = save_thm ("sg_13_thm", syll_rule2 sg_13);
val sg_14_thm = save_thm ("sg_14_thm", syll_rule2 sg_14);
```

The Square of Opposition

The Valid Syllogisms

The valid syllogisms have been described in Section 2.2.2.

All twenty four syllogisms are true under this semantics and have been proven. The actual theorems are shown in the theory listing in listing of theory *syllog4*[**?**].

To implement a mapping corresponding to the above semantics we must create a matching *mapkit* as follows:

SML

```
val mods_mapkit:mapkit = mkSimpMapkit ⌜:Term3⌝ [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝];
```

Proving the Syllogisms

The resulting translation yields goals which look exactly like the previous versions but have the meanings defined in this context. They are proven and stored in the theory listing (see Appendix **??**) by the following:

SML

```
proveSylls mods_mapkit syll_tac2 "" (syllogism_data1 @ syllogism_data2 @ syllogism_data3);
```

2.2.7 Existential Import in Universals

A more complete description of this interpretation, which is taken from Strawson [**?**] though not endorsed by him, is that existential import in relation both to the subject and the complement of the predicate, is present in universals and is absent from particular assertions. This combination is though by Strawson to be unsatisfactory primarily I believe because of its poor correspondence with any plausible account of the ordinary usage of the terms involved, however it appears also to be in poor correspondence with Aristotle. The difficulties can be traced to Strawson's acceptance of all four obversions, which appear not to have been endorsed by Aristotle. These obversions lead to the equivalence of contrapositives, and create a symmetry between subject and predicate in consequence of which presuppositions or implications of non emptyness attach both to subject and predicate.

My aim here is to confirm what Strawson says about this, which is that it satisfies all 14 laws and 28 syllogisms. He writes as if it were the only interpretation of this kind (i.e. not involving existential presuppositions rather than implications) which meet this requirement, on which I am sceptical but have not come to a definite conclusion.

SML

```
open_theory "aristotle";
force_new_theory "syllog5";
```

Semantics

Aristotle's syllogistic logic is concerned with inferences between judgements considered as predications. A predication in Aristotle affirms a *predicate* of some *subject*, but by contrast with more recent notions of predication the subject need not be an individual, the kinds of things which appear as predicates may also appear as subjects, and the relationship expressed seems closer to a modern eye to set inclusion than to what we now regard as predication. Since subject and predicate are for present purposes the same kind of thing, it is useful to have a name for that kind of thing, and I will use the name *property*.

There are four kinds of predication which we have here to account for, which we will do by offering definitions which provide a good model for syllogistic logic, i.e. one in which the syllogisms held to be true by Aristotle are in fact true. Before providing these definitions we must decide what kind of thing are the terms which are related by Aristotelian predication.

In HOL the most natural answer to this is "boolean valued functions" which are objects of type $\ulcorner : 'a \to BOOL \urcorner$ for some type of individuals which we can leave open by using the type variable $\ulcorner : 'a \urcorner$. This provides a simple model of Aristotle's non-u-p syllogistic reasoning. Four of the syllogisms which Aristotle considered valid fail under this conception of predicate, because among the objects of type $\ulcorner : 'a \to BOOL \urcorner$ is the function $\ulcorner \lambda x :' a \bullet F \urcorner$ which corresponds to a predicate with empty extension and does not admit inference from the universal to the existential (unless the universal is interpreted specially).

SML

```
declare_type_abbrev("Term2", [], ⌜:'a → BOOL⌝);
```

Predication

"o" is already in use for functional composition, so we will use "u" instead and then use an alias to permit us to write this as "o" (type inference will usually resolve any ambiguity).

To render these in HOL we first declare the relevant letters as infix operators:

They predication operators are defined as follows:

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : Term2 \to Term2 \to BOOL$$

$$\forall A\ B \bullet\ A\ a\ B \Leftrightarrow (\forall x \bullet\ A\ x \Rightarrow B\ x) \wedge (\exists x \bullet\ A\ x) \wedge (\exists x \bullet \neg\ B\ x)$$

HOL Constant

$\$\boldsymbol{e} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$

$\forall A\ B \bullet\ A\ e\ B \Leftrightarrow (\forall x \bullet\ A\ x \Rightarrow \neg\ B\ x) \wedge (\exists x \bullet\ A\ x) \wedge (\exists x \bullet\ B\ x)$

HOL Constant

$\$\boldsymbol{i} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$

$\forall A\ B \bullet\ A\ i\ B \Leftrightarrow (\exists x \bullet\ A\ x \wedge B\ x) \vee \neg\ (\exists x \bullet\ A\ x) \vee \neg\ (\exists x \bullet\ B\ x)$

HOL Constant

$\$\boldsymbol{u} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$

$\forall A\ B \bullet\ A\ u\ B \Leftrightarrow (\exists x \bullet\ A\ x \wedge \neg\ B\ x) \vee \neg\ (\exists x \bullet\ A\ x) \vee \neg\ (\exists x \bullet \neg\ B\ x)$

SML

```
declare_alias("o", ⌜$u⌝);
```

Note that as defined above these come in complementary pairs, a being the negation of o and e of i. If we had negation we could manage with just two predication operators.

The Laws of Immediate Inference

Though in the source of this kind of "literate script" are to be found the scripts for generating and checking the proofs of all the theorems presente, it will not be my practice to expose these scripts in the printed version of the document. These scripts are not usually intelligible other than in that intimate man-machine dialogue which they mediate, and sufficient knowledge for most purposes of the structure of the proof will be found in the detailed lemmas proven (since the level of proof automation is modest).

However, I will begin by exposing some of the scripts used for obtaining proofs of syllogisms in this model, to give the reader an impession of the level of complexity and kind of obscurity involved in this kind of formal work, I will not attempt sufficient explanation to make these scripts intelligible, they are best understood in the interactive environment, all the scripts are available for readers who want to run them.

Most readers are expected to skip over the gory details, the philosophical points at stake do not depend on the details of the proofs.

Before addressing the laws of immediate inference [6] I devise a tactic for automating simple proofs in this domain.

[6] in which I followed Strawson [?], though I can now cite Aristotle, Prior Analytic, Book 1, Part 2. [?]

The following elementary tactic expands the goal by applying the definitions of the operators and then invokes a general tactic for the predicate calculus. A rule is also defined using that tactic for direct rather than interactive proof.

SML

```
val syll_tac3 = asm_prove_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝]);
fun syll_rule3 g = tac_proof (g, syll_tac3);
```

SML

```
val e_conv_thm = save_thm("e_conv_thm", syll_rule3 ([⌜A e B⌝], ⌜B e A⌝));
val i_conv_thm = save_thm("i_conv_thm", syll_rule3 ([⌜A i B⌝], ⌜B i A⌝));
```

Simple Conversion The following two theorems are obtained.

```
e_conv_thm =  A e B ⊢ B e A
i_conv_thm =  A i B ⊢ B i A
```

The following two theorems show that the other obvious conversions are also false. Note that the theorems are not polymorphic, they are proven specifically for terms of type $\mathbb{N} \to BOOL$ (though any type of more than one element would do).

```
¬a_conv_thm = ⊢ ∃ A B• A a B ∧ ¬ B a A
¬o_conv_thm = ⊢ ∃ A B• A o B ∧ ¬ B o A
```

Conversion Per Accidens These come out OK.

SML

```
val ai_conv_thm = save_thm ("ai_conv_thm", syll_rule3([⌜A a B⌝], ⌜B i A⌝));
val eo_conv_thm = save_thm ("eo_conv_thm", syll_rule3([⌜A e B⌝], ⌜B u A⌝));
```

```
val ai_conv_thm = A a B ⊢ B i A : THM
val eo_conv_thm = A e B ⊢ B o A : THM
```

Obversion For these we need to define an operation of complementation on terms.

HOL Constant

```
Complement : Term2 → Term2
```

```
∀A α• (Complement A) α ⇔ ¬ (A α)
```

We will use "~" as a shorthand for "Complement".

SML
```
declare_alias ("∼", ⌜Complement⌝);
```

```
ae_obv_thm =    A a B ⊢ A e ∼ B
ea_obv_thm =    A e B ⊢ A a ∼ B
io_obv_thm =    A i B ⊢ A o ∼ B
oi_obv_thm =    A o B ⊢ A i ∼ B
```

Contraposition and Inversion

The Square of Opposition

This is complete with this semantics.

```
ao_contrad_thm = ⊢ A a B ⇔ ¬ A o B
ei_contrad_thm = ⊢ A e B ⇔ ¬ A i B
ae_contrar_thm = ⊢ ¬ (A a B ∧ A e B)
io_subcont_thm = ⊢ A i B ∨ A o B
ai_subalt_thm = ⊢ A a B ⇒ A i B
eo_subalt_thm = ⊢ A e B ⇒ A o B
```

The Syllogisms

First we make a *mapkit*.

SML
```
val s3mapkit:mapkit = mkSimpMapkit ⌜:Term2⌝ [⌜$a⌝,⌜$e⌝,⌜$i⌝,⌜$u⌝];
```

Then we apply this in generating and proving the syllogisms.

SML
```
proveGoals syll_tac3 "" (mkGoals s3mapkit syllogism_data1);
proveGoals syll_tac3 "" (mkGoals s3mapkit syllogism_data2);
proveGoals syll_tac3 "" (mkGoals s3mapkit syllogism_data3);
```

Three of the 24 generally accepted syllogisms prove unsound under this semantics: Camenes, Dimaris and Fresison.

$\vdash \neg\_Fresison = \vdash \neg\ (\forall\ P\ M\ \mathbb{S}\bullet\ P\ e\ M \wedge M\ i\ \mathbb{S} \Rightarrow \mathbb{S}\ o\ P)$
$\vdash \neg\_Dimaris = \vdash \neg\ (\forall\ P\ M\ \mathbb{S}\bullet\ P\ i\ M \wedge M\ a\ \mathbb{S} \Rightarrow \mathbb{S}\ i\ P)$
$\vdash \neg\_Camenes = \vdash \neg\ (\forall\ P\ M\ \mathbb{S}\bullet\ P\ a\ M \wedge M\ e\ \mathbb{S} \Rightarrow \mathbb{S}\ e\ P)$

for the record the counterexamples which disprove these syllogisms are all combinations of unversal (U) and empty (E) terms as follows.

	P	M	S
Fresison	U	E	U
Dimaris	E	U	U
Camenes	U	U	E

The theorems are also displayed in the theory listing in listing of theory *syllog5*[**?**]

2.2.8 Existential Import in Affirmations

This is my present best attempt at an interpretation which correponds closely to Aristotle.

SML
```
open_theory "aristotle";
force_new_theory "syllog6";
```

Semantics

Aristotle's syllogistic logic is concerned with inferences between judgements considered as predications. A predication in Aristotle affirms a *predicate* of some *subject*, but by contrast with more recent notions of predication the subject need not be an individual, the kinds of things which appear as predicates may also appear as subjects, and the relationship expressed seems closer to a modern eye to set inclusion than to what we now regard as predication. Since subject and predicate are for present purposes the same kind of thing, it is useful to have a name for that kind of thing, and I will use the name *property*.

There are four kinds of predication which we have here to account for, which we will do by offering definitions which provide a good model for syllogistic logic, i.e. one in which the syllogisms held to be true by Aristotle are in fact true. Before providing these definitions we must decide what kind of thing are the terms which are related by Aristotelian predication.

In HOL the most natural answer to this is "boolean valued functions" which are objects of type $\ulcorner : 'a \to BOOL \urcorner$ for some type of individuals which we can leave open by using the type variable $\ulcorner : 'a \urcorner$. This provides a simple model of Aristotle's non-u-p syllogistic reasoning. Four of the syllogisms which Aristotle considered valid fail under this conception of predicate, because among the objects of type $\ulcorner : 'a \to BOOL \urcorner$ is the function $\ulcorner \lambda x :' a \bullet F \urcorner$ which corresponds to a predicate with empty

extension and does not admit inference from the universal to the existential (unless the universal is interpreted specially).

SML

```
declare_type_abbrev("Term2", [], ⌜:'a → BOOL⌝);
```

Predication

"o" is already in use for functional composition, so we will use "u" instead and then use an alias to permit us to write this as "o" (type inference will usually resolve any ambiguity).

To render these in HOL we first declare the relevant letters as infix operators:

They predication operators are defined as follows:

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ a\ B \Leftrightarrow (\forall x \bullet\ A\ x \Rightarrow B\ x) \wedge \exists x \bullet\ A\ x$$

HOL Constant

$$\$\boldsymbol{e} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ e\ B \Leftrightarrow (\forall x \bullet\ A\ x \Rightarrow \neg\ B\ x)$$

HOL Constant

$$\$\boldsymbol{i} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ i\ B \Leftrightarrow (\exists x \bullet\ A\ x \wedge B\ x)$$

HOL Constant

$$\$\boldsymbol{u} : \mathit{Term2} \to \mathit{Term2} \to \mathit{BOOL}$$

$$\forall A\ B \bullet\ A\ u\ B \Leftrightarrow (\exists x \bullet\ A\ x \wedge \neg\ B\ x) \vee \neg\ (\exists x \bullet\ A\ x)$$

SML
```
declare_alias("o", ⌜$u⌝);
```

Note that as defined above these come in complementary pairs, a being the negation of o and e of i. If we had negation we could manage with just two predication operators.

The Laws of Immediate Inference

Though in the source of this kind of "literate script" are to be found the scripts for generating and checking the proofs of all the theorems presente, it will not be my practice to expose these scripts in the printed version of the document. These scripts are not usually intelligible other than in that intimate man-machine dialogue which they mediate, and sufficient knowledge for most purposes of the structure of the proof will be found in the detailed lemmas proven (since the level of proof automation is modest).

However, I will begin by exposing some of the scripts used for obtaining proofs of syllogisms in this model, to give the reader an impession of the level of complexity and kind of obscurity involved in this kind of formal work, I will not attempt sufficient explanation to make these scripts intelligible, they are best understood in the interactive environment, all the scripts are available for readers who want to run them.

Most readers are expected to skip over the gory details, the philosophical points at stake do not depend on the details of the proofs.

Before addressing the laws of immediate inference [7] I devise a tactic for automating simple proofs in this domain.

The following elementary tactic expands the goal by applying the definitions of the operators and then invokes a general tactic for the predicate calculus. A rule is also defined using that tactic for direct rather than interactive proof.

SML
```
val syll_tac6 = asm_prove_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝]);
fun syll_rule6 g = tac_proof (g, syll_tac6);
val syll_tac6b = REPEAT (POP_ASM_T ante_tac)
      THEN rewrite_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝])
      THEN contr_tac THEN asm_fc_tac[];
fun syll_rule6b g = tac_proof (g, syll_tac6b);
```

SML
```
val e_conv_thm = save_thm ("e_conv_thm", syll_rule6([⌜A e B⌝], ⌜B e A⌝));
val i_conv_thm = save_thm ("i_conv_thm", syll_rule6([⌜A i B⌝], ⌜B i A⌝));
```

[7] in which I followed Strawson [?], though I can now cite Aristotle, Prior Analytic, Book 1, Part 2. [?]

$val\ \mathbf{e\_conv\_thm} = A\ e\ B \vdash B\ e\ A : THM$
$val\ \mathbf{i\_conv\_thm} = A\ i\ B \vdash B\ i\ A : THM$

Simple Conversion

Conversion Per Accidens These are OK.

SML

$val\ ai\_conv\_thm = save\_thm\ (\texttt{"}ai\_conv\_thm\texttt{"},\ syll\_rule6([\ulcorner A\ a\ B \urcorner], \ulcorner B\ i\ A \urcorner));$
$val\ eo\_conv\_thm = save\_thm\ (\texttt{"}eo\_conv\_thm\texttt{"},\ syll\_rule6([\ulcorner A\ e\ B \urcorner], \ulcorner B\ u\ A \urcorner));$

$val\ \mathbf{ai\_conv\_thm} = A\ a\ B \vdash B\ i\ A : THM$
$val\ \mathbf{eo\_conv\_thm} = A\ e\ B \vdash B\ o\ A : THM$

Obversion For these we need to define an operation of complementation on terms.

HOL Constant

$\mathbf{Complement} : Term2 \to Term2$

$\forall A\ \alpha\bullet\ (Complement\ A)\ \alpha \Leftrightarrow \neg\ (A\ \alpha)$

We will use "~" as a shorthand for "Complement".

SML

$declare\_alias\ (\texttt{"}\sim\texttt{"}, \ulcorner Complement \urcorner);$

Only two of the obversions are valid.

$val\ ae\_obv\_thm = A\ a\ B \vdash A\ e \sim B : THM$
$val\ iu\_obv\_thm = A\ i\ B \vdash A\ o \sim B : THM$

The Square of Opposition

This is complete with this semantics.

$ao\_contrad\_thm = \vdash A\ a\ B \Leftrightarrow \neg\ A\ o\ B$
$ei\_contrad\_thm = \vdash A\ e\ B \Leftrightarrow \neg\ A\ i\ B$
$ae\_contrar\_thm = \vdash \neg\ (A\ a\ B \wedge A\ e\ B)$
$io\_subcont\_thm = \vdash A\ i\ B \vee A\ o\ B$
$ai\_subalt\_thm = \vdash A\ a\ B \Rightarrow A\ i\ B$
$eo\_subalt\_thm = \vdash A\ e\ B \Rightarrow A\ o\ B$

The Syllogisms

First we make a *mapkit*.

SML

```
val s6mapkit:mapkit = mkSimpMapkit ⌜:Term2⌝ [⌜$a⌝,⌜$e⌝,⌜$i⌝,⌜$u⌝];
```

Then we apply this in generating and proving the syllogisms.

SML

```
proveGoals syll_tac6 "" (mkGoals s6mapkit syllogism_data1);
proveGoals syll_tac6 "" (mkGoals s6mapkit syllogism_data2);
proveGoals syll_tac6 "" (mkGoals s6mapkit syllogism_data3);
```

The theorems are also displayed in the theory listing in listing of theory *syllog6*[**?**]

2.2.9 Modal Syllogisms

The language of syllogistic logic does not have operators over propositions. The only operators are the ones which apply predicates to subjects.

The modalities are perhaps therefore better thought of as kinds of judgements rather than as operations on propositions. This would give us three kinds of judgement, which assert a predication contingently, necessarily or possibly.

It is natural to consider the modal aspects in terms of possible worlds, and I will model it first in those terms (not knowing whether this will provide a good model of Aristotle's conception of modality). The propositional functions could then be modelled as functions from possible worlds to non-empty propositional functions.

The following first attempt at a modal syllogism is poorly representative of Aristotle, since, it appears, considered two definitions of possibilty in terms of necessity, but based his work on modal syllogisms primarily on one which does not correspond to the modern usage which is implicit in the treatment given here. It is to be expected therefore that any examination of the results obtained here will differ from Aristotle's views substantially because of this different conception of necessity. In due course I will add another kind of modal judgement which will be closer to the one principally investigated by Aristotle.

Aristotle's principle definition of 'possibly P' was:

> not necessarily P and not necessarily not P

This corresponds more closely with the concept 'contingent' or its modern counterpart, 'synthetic' than with contemporary usage of 'possible', though the contemporary rendering which corresponds to

Aristotle's other notion of possibility is a pre-Kripkean one in which we assume a fixed set of possible worlds with modal operators quantifying over the whole.

I adapt the treatment of u-p syllogisms by treating predicates as parameterised by a possible world.

Rather than using a type variable (which is what I did for the two preceding treatments) I will use two new type constants for individual aubstances and possible worlds.

It may suffice for the reader to refer directly to the theory listing in listing of theory *modsyllog*[**?**].

SML
```
open_theory "aristotle";
force_new_theory "modsyllog";
```

Semantics

The complexity required in the semantics of modal operators depends upon other features of the language in which they occur. Because the language of the syllogism is very simple, having neither propositional operators nor variables for individuals a very simple semantics may suffice. When we come to consider the metaphyics there will be some increase in the complexity of the other features of the language, and also a greater premium on getting the semantics to correspond intuitively with the content of the metaphysics, but at this stage we will adopt the simplest semantic model which seems likely to secure the results expressible in our restrictied language.

So are now talking about predicates parameterised by possible worlds. Furthermore, we will model this with a fixed set to individuals, independent of the possible world. Possible worlds differ only in the extension of predicates.

First some new types, "I" for individual substances, "W" for possible worlds:

SML
```
new_type ("I",0);
new_type ("W",0);
```

Then a type abbreviation for the predicates:

SML
```
declare_type_abbrev("MPROP", [], ⌜:W → I × (I → BOOL)⌝);
```

Predication

To work with this new type for the predicates we define a function which will convert this kind of predicate into the old kind, as follows:

HOL Constant

$$\boldsymbol{p} : MPROP \to (W \to I \to BOOL)$$

$$\forall A\bullet\ p\ A = \lambda w\ x\bullet\ let\ (v,f) = A\ w\ in\ x = v \vee f\ x$$

The following principle can be proven (proof omitted):

$$p\_\exists\_lemma =$$
$$\vdash \forall\ A\ w\bullet \exists\ v\bullet\ p\ A\ w\ v$$

This principle is what we need to prove the u-p syllogisms.

We then proceed in a similar manner to the other models, using the function p to convert the new kind of predicate into the old.

They predication operators are then defined. Note that the differences are small and uniform. The type $\ulcorner : Term3 \urcorner$ is changed to $\ulcorner : MPROP \urcorner$ and the function p is invoked before applying a predicate.

Now we think of a predication as being a set of possible worlds, or BOOLean valued function over possible worlds.

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : MPROP \to MPROP \to W \to BOOL$$

$$\forall A\ B\ w\bullet\ (A\ a\ B)\ w \Leftrightarrow \forall x\bullet\ p\ A\ w\ x \Rightarrow p\ B\ w\ x$$

HOL Constant

$$\$\boldsymbol{e} : MPROP \to MPROP \to W \to BOOL$$

$$\forall A\ B\ w\bullet\ (A\ e\ B)\ w \Leftrightarrow \forall x\bullet\ p\ A\ w\ x \Rightarrow \neg\ p\ B\ w\ x$$

HOL Constant

$$\$\boldsymbol{i} : MPROP \to MPROP \to W \to BOOL$$

$$\forall A\ B\ w\bullet\ (A\ i\ B)\ w \Leftrightarrow \exists x\bullet\ p\ A\ w\ x \wedge p\ B\ w\ x$$

HOL Constant

$\$\boldsymbol{u} : MPROP \to MPROP \to W \to BOOL$

$\forall A\ B\ w\bullet\ (A\ u\ B)\ w \Leftrightarrow \exists x\bullet\ p\ A\ w\ x \wedge \neg\ p\ B\ w\ x$

SML

$declare\_alias(\texttt{"}o\texttt{"}, \ulcorner\$u\urcorner);$

We now have to define some additional constants for the forms of judgement, which will assert the predications either of the actual world or of some or all possible worlds.

First I define a constant (rather loosely) to be the actual world:

HOL Constant

$\boldsymbol{actual\_world} : W$

T

Then the two modal judgement forms:

HOL Constant

$\Diamond : (W \to BOOL) \to BOOL$

$\forall s\bullet\ \Diamond\ s \Leftrightarrow \exists w\bullet\ s\ w$

HOL Constant

$\rangle\!\rangle : (W \to BOOL) \to BOOL$

$\forall s\bullet\ \rangle\!\rangle\ s \Leftrightarrow \forall w\bullet\ s\ w$

Aristotle's other notion of possibility is:

HOL Constant

$\Diamond_{\boldsymbol{a}} : (W \to BOOL) \to BOOL$

$\forall s\bullet\ \Diamond_a\ s \Leftrightarrow \neg\ (\forall w\bullet\ s\ w) \wedge \neg\ (\forall w\bullet\ \neg\ s\ w)$

Finally the non-modal judgements also need a judgement forming constant.

HOL Constant

$\vDash : (W \to BOOL) \to BOOL$

$\forall s\bullet \vDash s \Leftrightarrow s\ actual\_world$

Special difficulties are raised by reasoning with $\Diamond_a$ and to help with these difficulties it is useful to have negation as a kind of propositional operator in this modal logic.

SML

```
declare_prefix (350, "¬m");
```

HOL Constant

```
$¬m : (W → BOOL) → (W → BOOL)
```

```
∀x• ¬m x = λw• ¬ (x w)
```

Though this constant is distinct from the non-modal negation, we might as well drop the subscript where no ambiguity arises.

SML

```
declare_alias ("¬", ⌜$¬m⌝);
```

Laws of Immediate Inference

Before looking at the conversions there are some general rules which may be helpful for us though these probably are not in Aristotle.

$\Box\Diamond\_thm = \Box\ X \vdash \Diamond\ X$
$\Box\vDash\_thm = \Box\ X \vdash \vDash X$
$\vDash\Diamond\_thm = \vDash X \vdash \Diamond\ X$
$\Diamond_a\Diamond\_thm = \Diamond_a\ X \vdash \Diamond\ X$
$\Diamond_a\neg\Box\_thm = \Diamond_a\ X \vdash \neg \Box\ X$
$\Diamond_a\neg_m\_thm = \vdash \Diamond_a\ X \Leftrightarrow \Diamond_a\ (\neg_m\ X)$

$e\_conv\_thm = \vDash (A\ e\ B) \vdash \vDash (B\ e\ A)$
$i\_conv\_thm = \vDash (A\ i\ B) \vdash \vDash (B\ i\ A)$
$\Box e\_conv\_thm = \Box\ (A\ e\ B) \vdash \Box\ (B\ e\ A)$
$\Box i\_conv\_thm = \Box\ (A\ i\ B) \vdash \Box\ (B\ i\ A)$
$\Diamond e\_conv\_thm = \Diamond\ (A\ e\ B) \vdash \Diamond\ (B\ e\ A)$
$\Diamond i\_conv\_thm = \Diamond\ (A\ i\ B) \vdash \Diamond\ (B\ i\ A)$
$\Diamond_a\_e\_conv\_thm = \Diamond_a\ (A\ e\ B) \vdash \Diamond_a\ (B\ e\ A)$
$\Diamond_a\_i\_conv\_thm = \Diamond_a\ (A\ i\ B) \vdash \Diamond_a\ (B\ i\ A)$
$\Diamond_a\_ao\_conv\_thm = \vdash \Diamond_a\ (A\ a\ B) \Leftrightarrow \Diamond_a\ (A\ o\ B)$
$\Diamond_a\_ei\_conv\_thm = \vdash \Diamond_a\ (A\ e\ B) \Leftrightarrow \Diamond_a\ (A\ i\ B)$
$\Box\Diamond e\_conv\_thm = \Box\ (A\ e\ B) \vdash \Diamond\ (B\ e\ A)$
$\Box\vDash e\_conv\_thm = \Box\ (A\ e\ B) \vdash \vDash (B\ e\ A)$

Simple Conversion In this version of the semantics, "a" and "o" conversion is neither provable nor refutable. In the previous version (the one not admitting the u-p syllogisms), since the universe is a HOL type there is at least one individual, and contradictory predicates are allowed, we can use these two to disprove the two conversions. With this semantics there is no empty predicate, and we cannot know that there are two distinct predicates.

$\models ai\_conv\_thm = \models (A\ a\ B) \vdash \models (B\ i\ A)$
$\models eo\_conv\_thm = \models (A\ e\ B) \vdash \models (B\ o\ A)$
$\Vdash\!\!\Diamond ai\_conv\_thm = \quad \Vdash (A\ a\ B) \vdash \Diamond (B\ i\ A)$
$\Vdash\!\!\Diamond eo\_conv\_thm = \quad \Vdash (A\ e\ B) \vdash \Diamond (B\ o\ A)$

Conversion Per Accidens

The Valid Modal Syllogisms

The valid syllogisms have been described in Section 2.2.2.

All nineteen syllogisms supposed valid by Aristotle are true under this semantics and have been proven. A further five[8] have also been proven, giving a total of 24. When combinations of modal operators are added to this the number gets quite large, so, rather than proving all the valid cases I will prove sufficient to enable the rest to be automatically proven.

This will involve some theorems which are not strictly syllogistic.

The actual theorems proved are shown in the theory listing in listing of theory *modsyllog*[**?**].

Because of the modal operators the generation of the syllogisms is more complicated. The generation functions are adapted to allow a single modal operator to be applied to each of the premises and the conclusion.

SML

```
fun mk_pred q s p = mk_app(mk_app (q, s), p);

fun mk_syll vt (a,b,c,d) (q1, q2, q3) =
      ([mk_pred q1 a b, mk_pred q2 c d],
            mk_pred q3 (mk_var("𝕊", vt)) (mk_var("P", vt)));

fun mk_relt t = mk_ctype ("→", [t, mk_ctype ("→", [t, ⌜:BOOL⌝])]);

fun mk_syllp vt (s, n) =
      mk_syll vt (nth (n−1) (figurest vt)) (optrip_from_text (mk_relt vt) s);
```

[8] Which I got from Strawson [**?**].

SML

```
fun syll_prove msp suff tac (a,n) =
    let val thm = tac_proof (msp (a,n), tac) handle _ => t_thm
    in (concat [a, suff], thm)
    end;

fun syll_prove_and_store msp suff tac (a,n) =
    let val res = syll_prove msp suff tac (a,n);
        val _ = save_thm res
    in res
    end;
```

SML

```
fun map_goal f (st, t) = (map f st, f t);

fun mk_modt vt = mk_ctype ("→", [vt,
    mk_ctype ("→", [vt, (mk_ctype ("→", [⌜:W⌝, ⌜:BOOL⌝]))])]);

fun mk_modsyll vt (s, n) =
    mk_syll vt (nth (n−1) (figurest vt)) (optrip_from_text (mk_modt vt) s);

fun modgoal (mo1, mo2, mo3) ([p1,p2], c) =
    ([mk_app (mo1, p1), mk_app (mo2, p2)], mk_app (mo3, c));

fun mk_modsyllp mot p = modgoal mot (mk_modsyll ⌜:MPROP⌝ p);
```

This defines the function *mk_modsyllp* whose type is shown:

```
val mk_modsyllp = fn: TERM * TERM * TERM -> string * int -> TERM list * TERM
```

in which the *TERM* parameters are modal operators the next argument is a pair consisting of a string which is the name of a syllogism and a number which is the number of the figure. The result is a goal for proof.

An example of its use is:

SML

```
mk_modsyllp (⌜⦀⌝,⌜⊨⌝,⌜◇⌝) ("Barbara", 1);
```

which yields:

```
val it = ([⌜⦀ (M a P)⌝, ⌜⦀ (𝕊 a M)⌝], ⌜⦀ (𝕊 a P)⌝) : TERM list * TERM
```

General Results

The logic of the modal operators is completely independent of the logic of the syllogism. The relevant results can be stated and proven in HOL concisely, but these statements are not in the language of the syllogism.

There are in effect just seven modal truths, each of which appears in 24 forms, one for each of the valid non-modal syllogisms.

Rather than proving all 192 theorems (counting the non-modal truths in this modal model), I prove the eight proformas expressed in HOL. From these eight all 192 theorems can be obtained by proving (a special form of) one of the valid syllogisms and instantiating one of the general modal rules using it.

I omit the details of the metalanguage scripts which automate all this.

The following lists the valid modal forms. In each tuple the three entries give the modalities of the first and second premise and the conclusion respectively. Taking any valid syllogism and applying modal operators using one of the patterns in this table will give a valid modal syllogism.

SML

```
val mod_gen_params =
      [(⌜⟫⌝, ⌜⟫⌝, ⌜⟫⌝),
       (⌜⟫⌝, ⌜⟫⌝, ⌜◇⌝),
       (⌜⟫⌝, ⌜⟫⌝, ⌜⊨⌝),
       (⌜◇⌝, ⌜⟫⌝, ⌜◇⌝),
       (⌜⟫⌝, ⌜◇⌝, ⌜◇⌝),
       (⌜⟫⌝, ⌜⊨⌝, ⌜⊨⌝),
       (⌜⊨⌝, ⌜⟫⌝, ⌜⊨⌝),
       (⌜⊨⌝, ⌜⊨⌝, ⌜⊨⌝)];
```

The set of general HOL theorems which facilitate the proofs of these modal syllogisms is as follows:

ProofPower Theorems

```
val mod_gen_thms =
  [⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⟫ FP ∧ ⟫ SP ⇒ ⟫ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⟫ FP ∧ ⟫ SP ⇒ ◇ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⟫ FP ∧ ⟫ SP ⇒ ⊨ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ◇ FP ∧ ⟫ SP ⇒ ◇ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⟫ FP ∧ ◇ SP ⇒ ◇ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⟫ FP ∧ ⊨ SP ⇒ ⊨ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⊨ FP ∧ ⟫ SP ⇒ ⊨ CS,
    ⊢ ∀ FP SP CS• (∀ w• FP w ∧ SP w ⇒ CS w) ⇒ ⊨ FP ∧ ⊨ SP ⇒ ⊨ CS]
: THM list
```

In the above theorems the variables *FP*, *SP*, *CS*, stand respectively for *first premise*, *second premise*, *conclusion of syllogism* and range over modal propositions (which have type ⌜:W → BOOL⌝).

Proving the Syllogisms

I then prove the 24 non-modal syllogisms in the required form and infer forward using the above 8 theorems to obtain a total of 192 theorems true in this model of the modal syllogism.

Details of scripts omitted.

The automated proof the yields the expected 192 modal syllogisms, of which we display only the first few (and do not save them in the theory):

```
val valid_G_modsylls =
   [⊨ (P a M), ⊨ (M e 𝕊) ⊢ ⊨ (𝕊 o P), ⊨ (P a M), ⫦ (M e 𝕊) ⊢ ⊨ (𝕊 o P),
      ⫦ (P a M), ⊨ (M e 𝕊) ⊢ ⊨ (𝕊 o P), ⫦ (P a M), ◇ (M e 𝕊) ⊢ ◇ (𝕊 o P),
      ◇ (P a M), ⫦ (M e 𝕊) ⊢ ◇ (𝕊 o P), ⫦ (P a M), ⫦ (M e 𝕊) ⊢ ⊨ (𝕊 o P),
      ⫦ (P a M), ⫦ (M e 𝕊) ⊢ ◇ (𝕊 o P), ⫦ (P a M), ⫦ (M e 𝕊) ⊢ ⫦ (𝕊 o P),
      ⊨ (P a M), ⊨ (𝕊 e M) ⊢ ⊨ (𝕊 o P), ⊨ (P a M), ⫦ (𝕊 e M) ⊢ ⊨ (𝕊 o P),
      ⫦ (P a M), ⊨ (𝕊 e M) ⊢ ⊨ (𝕊 o P), ⫦ (P a M), ◇ (𝕊 e M) ⊢ ◇ (𝕊 o P),
      ◇ (P a M), ⫦ (𝕊 e M) ⊢ ◇ (𝕊 o P), ⫦ (P a M), ⫦ (𝕊 e M) ⊢ ⊨ (𝕊 o P),
      ⫦ (P a M), ⫦ (𝕊 e M) ⊢ ◇ (𝕊 o P), ⫦ (P a M), ⫦ (𝕊 e M) ⊢ ⫦ (𝕊 o P),
      ⊨ (P e M), ⊨ (𝕊 a M) ⊢ ⊨ (𝕊 o P), ⊨ (P e M), ⫦ (𝕊 a M) ⊢ ⊨ (𝕊 o P),
      ⫦ (P e M), ⊨ (𝕊 a M) ⊢ ⊨ (𝕊 o P), ⫦ (P e M), ◇ (𝕊 a M) ⊢ ◇ (𝕊 o P),
...
```

```
SML
length valid_G_modsylls;

val it = 192 : int
```

2.2.10 Demonstrative Truth

An important part of Aristotle's philosophy is his concept of demonstrative science.

A proof is demonstrative if it proceeds from first principles and is deductively sound. Truths established in this way are necessary because the first principles must be essential and hence necessary and sound deduction preserves necessity.

If we understand Hume's "intuitively certain" as a reference to the criteria for Aristotle's first principles, and understand Hume as using the term 'demonstrative' in the same sense as Aristotle, then it is plausible that Hume's "truths of reason" are the same as Aristotle's truths of demonstrative science.

It is tempting to use the word demonstrative for 'truths of reason' though in Aristotle and Hume the first principles do not count as demonstrable. This follows modern logic in using concepts such as 'theorem' and 'valid sentence' which apply to logical axioms as well as results deduced from them. It is tempting also to identify these concepts with the concept of analyticity. This last point is aided by the connection in Aristotle between essential truth (which must be posessed by the first principles) and definition, which seems close, and which distinguishes his accidental predications from essential predications. A weak point in this is the question of the first principles of the various sciences, and the doubts one can reasonably have about whether Aristotle's essential truth must be analytic.

However, we now expect deductive systems to be incomplete, and hence that not all analytic propositions are provable. However, the incompleteness may arise from adopting a fixed set of first principles, rather from incompleteness of deduction. In this case, the availability of an open set of first principles makes completeness in principle possible. To sustain this principle in relation to set theory, for example, we would have to regard ourselves as having some definition of the concept of set relative to which the present axioms (say those of ZFC) are essential, and relative to which extensions as necessary to prove progressively more difficult results can also be seen to be essential. This is not entirely implausible. This is close to the rationale for large cardinal axioms. The informal description of the cumulative hierarchy as the domain of set theory involves the idea that the construction of well-founded sets from other well-founded sets of lesser rank proceeds indefinitely, and hence any axiom which states that a set of a certain rank exists must be true.

Does formal modelling contribute anything to this discussion?

The above discussion involves ideas which belong to Aristotle's metaphysics rather than his logic. So a fuller formal analysis of these ideas will have to wait until we get to the Metaphysics.

However some aspects may be considered here. For example, we need to know that from necessary premises only necessary premises are derivable by syllogisms.

2.3 Metaphysics (II)

In this section I offer a single model integrating Modal Syllogisms with the distinction between essential and accidental predication.

My interest is primarily in the extent to which may be found in Aristotle's philosophy a precursor of Hume's fork or the modern distinctions between analytic and synthetic or necessary and contingent propositions. I see three Aristotelian ideas which have some relevance.

- the distinction between necessary and contingent propositions
- the distinction between essential and accidental predication
- the notion of demonstrability

It is only when we combine the syllogism with the metaphyics that we can explore the relationship between these various concepts.

I will give higher priority in this model to good structure while remaining faithful to Aristotle. It is not the purpose of this model to further investigate the position in relation to Aristotle of Grice, Codd or Speranza.

I have constructed this model to give a good correspondence between necessary truth and essential predication. If the model is successful in that respect it remains to consider whether it is consistent with the philosophy of Aristotle. I do not know whether Aristotle talked about the relationship between essence and necessity.

I also hope that the model may help to explore the question of whether demonstrable truth, or rather the truths which are either "intuitively or demonstrably certain" to use Hume's words, coincides with necessary truth.

SML

```
open_theory "aristotle";
force_new_theory "syllmetap";
```

2.3.1 Semantics

In my first metaphysical model the main question in relations to subject matter was "what are subjects and predicates", to which a model of Aristotelian categories gives an answer. The introduction of modality makes it necessary to consider something like possible worlds. These were left uninterpreted in the modal treatment of the syllogism, but now that we expose the distinction between essential and accidental predication is it desirable to identify possible worlds with that which is accidental.

What is accidental is the extension of individual attributes, and this gives our concept of possible world.

It is convenient at this point to consider the question of extensionality.

According to Grice/Codd/Speranza, essential predicates are extensional:

$$A\ izz\ B\ \wedge\ B\ izz\ A\ \Rightarrow\ A\ =\ B$$

but I know no reason to suppose that accidental predication is, and it seems counter-intuitive that it should be. Consequently the modelling of an accidental predicate using a BOOLean valued function in HOL (in which functions are extensional) is inappropriate. So I will separate out the extension from the individual attribute.

Individuals belong to categories, and are collected together in groups which determine the nature of essential predication within that category. The simplest way of getting the right structure is to use some type of tags to differentiate individuals within a category, using the same collection of tags in each category (with the unintended effect of ensuring that each category has the same number of individuals, which I hope will have a significant effect on the resulting theory). The individuals are then represented by an ordered pair consisting of a category and a tag.

For most purposes the number of categories is not important, though we must have a category of substances, so a completely undifferentiated new type might have sufficed. However, it turns out that some things don't work, and its useful to have at least one non-substantial category in order to prove that they don't work. So I introduce a new type of non-substantial (attribute) categories (so we get at least one) and then make the type of categories by adding one more.

SML

```
new_type ("ACAT", 0);
new_type ("TAG", 0);
```

SML

```
declare_type_abbrev("CAT", [], ⌜:ONE+ACAT⌝);
```

One of the categories will be the category of substance, it doesn't matter which one but we might as well use the odd One on the left of the sum (so you can test for substance using *IsL*).

HOL Constant

```
Category_of_Substance : CAT
```

```
Category_of_Substance = InL One
```

An individual will therefore be modelled as an ordered pair consisting of a tag and a category. This is captured by the following type abbreviation.

SML

```
declare_type_abbrev ("I", [], ⌜:CAT × TAG⌝);
```

A possible world is then an assignment of extensions to individual attributes, where an extension is a set of particulars. Since we only want individual attributes, we do not use type *I* for the domain. Particulars always belong to the category of substance, so we only need a set of tags in the result, the category is implicit.

SML

```
declare_type_abbrev ("W", [], ⌜:ACAT × TAG → TAG SET⌝);
```

This does include an assignement of extensions to particulars, but this plays no role, only intefering with the identity criteria for possible worlds, which do not feature in the theory.

I need to distinguish one possible world which is the actual world:

HOL Constant

```
actual_world : W
```

```
T
```

Since the individuals are pairs it might be handy to have appropriately named projection funtions which extract the two components:

HOL Constant

$$\begin{array}{|l} \boldsymbol{category} : I \to CAT; \\ \boldsymbol{tag} : I \to TAG \\ \hline \forall ct \bullet\ category\ ct = Fst\ ct \\ \quad \wedge\ \ tag\ ct = Snd\ ct \end{array}$$

Finally the question of what subjects and predicates are can be determined. I will call them *TermM*s and they are either a set of particulars or a set of attributes. To allow for complementation I add a boolean component, which if true indicates a complement.

The sets in this case must be non-empty if we are to retain the u-p syllogisms (in default of a different universal). The method used in my model of the modal syllogism in Section 2.2.9 will not do here, because (at least according to Code) we need an extensionality result, so I have instead introduced [9] for this purpose a new type (*NESET*) of non-empty sets which will give us both the u-p syllogisms and extensionality (for essential predication, not for accidental predication).

A term is therefore modelled as either a non-empty set of individual substances or a non-empty set of individual attributes.

SML

$$declare\_type\_abbrev\ (\texttt{"}\boldsymbol{TermM}\texttt{"},\ [],\ \ulcorner{:}(CAT \times TAG\ NESET)\urcorner);$$

HOL Constant

$$\begin{array}{|l} \boldsymbol{mk\_SubsTerm} : TAG\ \mathbb{P} \to TermM \\ \hline \forall s \bullet\ mk\_SubsTerm\ s = (Category\_of\_Substance,\ NeSet\ s) \end{array}$$

HOL Constant

$$\begin{array}{|l} \boldsymbol{mk\_AttrTerm} : CAT \times TAG\ \mathbb{P} \to TermM \\ \hline \forall s \bullet\ mk\_AttrTerm\ s = (Fst\ s,\ NeSet\ (Snd\ s)) \end{array}$$

It may be useful to have a name for the predicate encompassing all substance.

HOL Constant

$$\begin{array}{|l} \boldsymbol{Substance} : TermM \\ \hline Substance = mk\_SubsTerm\ Universe \end{array}$$

[9] The definition has been placed in a separate document, [**?**].

2.3.2 Predication

The syllogism comes with four kinds of predication (a, e, i, o), and the metaphysics with three (izz, hazz and izz or hazz). Combining these would give twelve combinations.

To simplify a bit I will separate out the quantifier but defining *All* and *Some* appropriately, and provide a postfix negator for izz an hazz.

I will then treat the modal operators as operators over propositions, and introduce the syllogism as a kind of judgement.

The type of the primitive copulas is:

SML

```
declare_type_abbrev("COPULA", [], ⌜:I → TermM → (W → BOOL)⌝);
```

The first parameter is an individual substance or attribute rather than a TermM, the quantifying operator will arrange for each of the relevant individuals or attributes to be supplied.

SML

```
declare_type_abbrev ("MPROP", [], ⌜:W → BOOL⌝);
```

Propositions

Complementation The distinction between affirmative and negative is achieved by a postfix negation so we can say "izz not", "hazz not" or "are not".

SML

```
declare_postfix (100, "not");
```

HOL Constant

```
$not : COPULA → COPULA
```

```
∀pred• pred not = λpa t w• ¬ pred pa t w
```

Quantifiers The following function is used by both quantifiers to check if something is in the range of quantification.

Think of a TermM as denoting a set of individuals, this is a test for membership of that set. The complications are because substances and attributes have different types in this model.

SML

```
declare_infix(300, "InTermM");
```

HOL Constant

$$\$\boldsymbol{InTermM} : I \to TermM \to BOOL$$

$$\forall\ c1t\ c2ts \bullet c1t\ InTermM\ c2ts \Leftrightarrow Fst\ c1t = Fst\ c2ts \wedge Snd\ c1t \in PeSet\ (Snd\ c2ts)$$

$$interm\_\exists\_lemma =$$
$$\vdash \forall\ t\bullet \exists\ j\bullet\ j\ InTermM\ t$$

We then use that membership test in defining the quantifiers. The quantifiers expect to be supplied with a copula and a term. The quantifier then predicates using the copula the term of everything or something in the domain of quantification (which is the subject term). The copulas are defined below.

HOL Constant

$$\boldsymbol{All} : TermM \to (I \to TermM \to MPROP) \to TermM \to MPROP$$

$$\forall\ s\ r\ p\bullet All\ s\ r\ p = \lambda w\bullet \forall z\bullet\ z\ InTermM\ s \Rightarrow r\ z\ p\ w$$

HOL Constant

$$\boldsymbol{Some} : TermM \to (I \to TermM \to MPROP) \to TermM \to MPROP$$

$$\forall\ s\ r\ p\bullet Some\ s\ r\ p = \lambda w\bullet \exists z\bullet\ z\ InTermM\ s \wedge r\ z\ p\ w$$

Predicators For essential predication it is necessary that the individual and the predicate are both of the same category and then reduces under our model to set membership. In effect. since the non-substantial individuals are tagged with their category, we need only deal separately with the distinction between substantial and non-substantial and the set inclusion will ensure a match in the non-substantial categories.

HOL Constant

$$\boldsymbol{izz} : I \to TermM \to MPROP$$

$$\forall\ j\ t\bullet izz\ j\ t = \lambda w\bullet\ j\ InTermM\ t$$

For accidental predication the subject term must be substantial and the predicate may not be. We then need some member of the predicate to be attributable to the substance.

HOL Constant

$$\boldsymbol{hazz} : I \to TermM \to MPROP$$

$$\forall\ c1t\ c2ts\bullet\ hazz\ c1t\ c2ts = \lambda w\bullet$$
$$\quad Fst\ c1t = Category\_of\_Substance$$
$$\wedge\quad \neg\ Fst\ c2ts = Category\_of\_Substance$$
$$\wedge\quad (\exists b\bullet\ b \in PeSet\ (Snd\ c2ts) \wedge (Snd\ c1t) \in w\ (OutR(Fst\ c2ts),\ b))$$

$$not\_izz\_and\_hazz\_lemma1 =$$
$$\quad \vdash \forall\ pa\ t\ w\bullet \neg\ (izz\ pa\ t\ w \wedge hazz\ pa\ t\ w)$$

HOL Constant

$$\boldsymbol{are} : I \to TermM \to MPROP$$

$$\forall\ pa\ t\bullet\ are\ pa\ t = \lambda w\bullet\ izz\ pa\ t\ w \vee hazz\ pa\ t\ w$$

$$\boldsymbol{are\_izz\_neq\_hazz\_lemma} =$$
$$\quad \vdash \forall\ pa\ t\ w\bullet\ are\ pa\ t\ w \Leftrightarrow \neg\ (izz\ pa\ t\ w \Leftrightarrow hazz\ pa\ t\ w)$$

$$\boldsymbol{All\_are\_izz\_or\_hazz\_lemma} =$$
$$\quad \vdash \forall\ A\ B\ w\bullet\ All\ A\ are\ B\ w \Leftrightarrow All\ A\ izz\ B\ w \vee All\ A\ hazz\ B\ w$$

$$\boldsymbol{Some\_are\_izz\_or\_hazz\_lemma} =$$
$$\quad \vdash \forall\ A\ B\ w\bullet\ Some\ A\ are\ B\ w \Leftrightarrow Some\ A\ izz\ B\ w \vee Some\ A\ hazz\ B\ w$$

$$\boldsymbol{All\_are\_not\_lemma} =$$
$$\quad \vdash \forall\ A\ B\ w\ \bullet\ All\ A\ (are\ not)\ B\ w \Leftrightarrow All\ A\ (izz\ not)\ B\ w \wedge All\ A\ (hazz\ not)\ B\ w$$

Modal Operators In this model the model operators are operators over propositions.

HOL Constant

$$\Diamond : MPROP \to MPROP$$

$$\forall p\bullet\ \Diamond\ p = \lambda w\bullet\ \exists w'\bullet\ p\ w'$$

HOL Constant

$$\rangle\!\rangle : MPROP \to MPROP$$

$$\forall p\bullet\ \rangle\!\rangle\ p = \lambda w\bullet\ \forall w'\bullet\ p\ w'$$

2.3.3 Propositional Operators

Though the truth functional propositional operators do not feature in the syllogism it is nevertheless useful to have them in giving a full account of Aristotle's logic and they are therefore here defined.

That these propositional operators are "truth functional", in a context in which propositions are not regarded as denoting truth values requires a little explanation perhaps. Our propositions are families of truth values indexed by possible worlds, i.e. functions from possible worlds to truth values, or in the context of a two valued logic (which Aristotle's seems to be), sets of possible worlds. In this context the usual truth functional operators can be expressed by mapping the usual operator over the set of possible worlds, i.e. the result in every possible world is the result of applying the truth functional operator to the values of the propositions in that possible world. These also correspond to the obvious set theoretic operation if the propositions are thought of as sets of possible worlds, i.e. intersetion for conjunction, complementation for negation.

The symbols for the operators are already in use, so we define the operations using decorated variants of the symbols and use an alias to allow the undecorated symbol to be used.

HOL Constant

$$\neg_{\boldsymbol{a}} : MPROP \to MPROP$$

$$\forall p\bullet\ \neg_a\ p = \lambda w\bullet\ \neg\ (p\ w)$$

SML

```
declare_alias ("¬", ⌜¬a⌝);
```

SML

```
declare_infix(220, "∧a");
```

HOL Constant

$$\$\wedge_{\boldsymbol{a}} : MPROP \to MPROP \to MPROP$$

$$\forall p\ q\bullet\ (p\ \wedge_a\ q) = \lambda w\bullet\ (p\ w)\ \wedge\ (q\ w)$$

SML

```
declare_alias ("∧", ⌜$∧a⌝);
```

SML

```
declare_infix(210, "⇒a");
```

HOL Constant

$\$\Rightarrow_{\boldsymbol{a}} : MPROP \to MPROP \to MPROP$

$\forall p\ q\bullet\ (p \Rightarrow_a q) = \lambda w\bullet\ p\ w \Rightarrow q\ w$

SML

$declare\_alias\ (\texttt{"}\Rightarrow\texttt{"},\ \ulcorner\$\Rightarrow_a\urcorner)$;

SML

$declare\_infix(200,\ \texttt{"}\Leftrightarrow_a\texttt{"})$;

HOL Constant

$\$\Leftrightarrow_{\boldsymbol{a}} : MPROP \to MPROP \to MPROP$

$\forall p\ q\bullet\ (p \Leftrightarrow_a q) = \lambda w\bullet\ p\ w \Leftrightarrow q\ w$

SML

$declare\_alias\ (\texttt{"}\Leftrightarrow\texttt{"},\ \ulcorner\$\Leftrightarrow_a\urcorner)$;

2.3.4 Quantification

The Grice/Code analysis makes use of quantifiers, particularly existential quantification. To verify the formulae in this context we therefore need to define modal version of the quantifiers.

SML

$declare\_binder\ \texttt{"}\forall_a\texttt{"}$;

HOL Constant

$\$\forall_{\boldsymbol{a}} : (TermM \to MPROP) \to MPROP$

$\forall mpf\bullet\ \$\forall_a\ mpf = \lambda w\bullet\ \forall t\bullet\ mpf\ t\ w$

SML

$declare\_alias\ (\texttt{"}\forall\texttt{"},\ \ulcorner\$\forall_a\urcorner)$;

SML

$declare\_binder\ \texttt{"}\exists_a\texttt{"}$;

HOL Constant

$$\$\exists_{\boldsymbol{a}} : (TermM \rightarrow MPROP) \rightarrow MPROP$$

$$\forall mpf\bullet\ \$\exists_a\ mpf\ =\ \lambda w\bullet\ \exists t\bullet\ mpf\ t\ w$$

SML

$$declare\_alias\ (\texttt{"}\exists\texttt{"},\ \ulcorner\$\exists_a\urcorner);$$

2.3.5 Judgements

I'm not yet clear what to offer here, so for the present I will define two kinds of sequent, which will be displayed with the symbols $\models$ asnd $\Vdash$. the former being a kind of contingent material implication and the latter a necessary implication.

Both form of judgement seem suitable for expressing the rules of the syllogism at first glance but which can also be used for conversions.

The first expresses a contingent entailment, that if some arbitrary finite (possibly empty) collection of premises are contingently true, then some conclusion will also be true. Since the consequence is material, and the premisses might be contingent, the conclusion might also be contingent. One might hope that if the rules of the syllogism are applied and the premises are necessary, then so will be the conclusions.

SML

$$declare\_infix(100,\ \texttt{"}\models\texttt{"});$$

HOL Constant

$$\$\models : MPROP\ LIST \rightarrow MPROP \rightarrow BOOL$$

$$\forall lp\ \ c\bullet\ lp \models c \Leftrightarrow Fold\ (\lambda p\ t\bullet\ p\ actual\_world \wedge t)\ lp\ T \Rightarrow c\ actual\_world$$

This one says that in every possible world the premises entail the conclusion (still material).

SML

$$declare\_infix(100,\ \texttt{"}\Vdash\texttt{"});$$

HOL Constant

$$\$\Vdash : MPROP\ LIST \rightarrow MPROP \rightarrow BOOL$$

$$\forall lp\ \ c\bullet\ lp \Vdash c \Leftrightarrow \forall w\bullet\ Fold\ (\lambda p\ t\bullet\ p\ w \wedge t)\ lp\ T \Rightarrow c\ w$$

In the present context the choice between the two is probably immaterial, since we know no more about the actual world than any other, so anything that we can prove to be true contingently, we can also prove to be true necessarily.

2.3.6 Conversions

Premisses, their Modes and Conversions See: Prior Analytics Book 1 Part 2 Paragraph 2.

> First then take a universal negative with the terms A and B.
>
> If no B is A, neither can any A be B. For if some A (say C) were B, it would not be true that no B is A; for C is a B.
>
> But if every B is A then some A is B. For if no A were B, then no B could be A. But we assumed that every B is A.
>
> Similarly too, if the premiss is particular. For if some B is A, then some of the As must be B. For if none were, then no B would be A. But if some B is not A, there is no necessity that some of the As should not be B; e.g. let B stand for animal and A for man. Not every animal is a man; but every man is an animal.

These work out fine for *izz*, so I will do those first, and then show that they fail for *hazz* and *is*.

The first and third conversions are most useful when expressed as an equation, since our proof system is based primarily on rewriting using equations.

```
izz_not_lemma =
    ⊢ All B (izz not) A = All A (izz not) B
some_izz_lemma =
    ⊢ Some B izz A = Some A izz B
```

These we also supply as our Aristotelian judgements, together with the second which does not give an equation. The second conversion embodies the u-p syllogisms.

```
izz_conv1 = ⊢
    [All B (izz not) A] ⊩ All A (izz not) B

izz_conv2 = ⊢
    [All B izz A] ⊩ Some A izz B

izz_conv3 = ⊢
    [Some B izz A] ⊩ Some A izz B
```

Now we look at *hazz*.

The following theorems state that the two equational conversions are both false for *hazz*.

$$\begin{array}{l}\boldsymbol{not\_hazz\_not\_lemma} = \\ \quad \vdash \neg\ (\forall\ A\ B\bullet\ All\ B\ (hazz\ not)\ A = All\ A\ (hazz\ not)\ B)\end{array}$$

$$\begin{array}{l}\boldsymbol{not\_some\_hazz\_lemma} = \\ \quad \vdash \neg\ (\forall\ A\ B\bullet\ Some\ B\ hazz\ A = Some\ A\ hazz\ B)\end{array}$$

Aristotle's second conversion also fails for *hazz*, because in incorporates an application of the third in effect. If we simplify by removing the final flip we get:

$$\begin{array}{l}\boldsymbol{hazz\_conv2} = \\ \quad \vdash [All\ A\ hazz\ B] \Vdash Some\ A\ hazz\ B\end{array}$$

Since *is* is the conjunction of *izz* and *hazz* it is likely that it would yeild similar results to *hazz*.

2.3.7 Modal Conversions

Prior Analytics Book 1 Part 3 See: Universal and Possible Premisses and their Conversions.

These are the conversions in relation to necessity and possibility described by Aristotle:

1. If it is necessary that no B is A, it is necessary also that no A is B.
2. If all or some B is A of necessity, it is necessary also that some A is B.
3. If it is possible that all or some B is A, it will be possible that some A is B.
4. and so on

So in this section Aristotle only offers variants of the previous conversions with either "possible" or "necessary" attached to both premiss and conclusion.

We can prove generally that modal operators can be introduced into a conversion:

$$\begin{array}{l}\boldsymbol{\Diamond\_conv} = \\ \quad \vdash [P] \Vdash Q \Rightarrow [\Diamond\ P] \Vdash \Diamond\ Q\end{array}$$

$$\begin{array}{l}\boldsymbol{\rangle\!)\_conv} = \\ \quad \vdash [P] \Vdash Q \Rightarrow [\rangle\!)\ P] \Vdash \rangle\!)\ Q\end{array}$$

$\Vert\!\!\rangle\_izz\_thm = \vdash [\Vert\!\!\rangle\ (All\ A\ izz\ B)] \vDash All\ A\ izz\ B$
$\Vert\!\!\rangle\_hazz\_thm = \vdash [\Vert\!\!\rangle\ (All\ A\ hazz\ B)] \vDash All\ A\ izz\ B$
$izz\_\Vert\!\!\rangle\_thm = \vdash [All\ A\ izz\ B] \vDash \Vert\!\!\rangle\ (All\ A\ izz\ B)$
$not\_\Vert\!\!\rangle\_hazz\_thm = \vdash [] \vDash (\neg\ \Vert\!\!\rangle\ (All\ A\ hazz\ B))$

$\Vert\!\!\rangle\_izz\_thm2 = \vdash [\Vert\!\!\rangle\ (All\ A\ izz\ B)] \Vdash All\ A\ izz\ B$
$\Vert\!\!\rangle\_hazz\_thm2 = \vdash [\Vert\!\!\rangle\ (All\ A\ hazz\ B)] \Vdash All\ A\ izz\ B$
$izz\_\Vert\!\!\rangle\_thm2 = \vdash [All\ A\ izz\ B] \Vdash \Vert\!\!\rangle\ (All\ A\ izz\ B)$
$not\_\Vert\!\!\rangle\_hazz\_thm2 = \vdash [] \Vdash (\neg\ \Vert\!\!\rangle\ (All\ A\ hazz\ B))$

$\Vert\!\!\rangle\_hazz\_thm$ is a bit odd. Really what I wanted to prove was that no accidental predication is necessary, but I have no negation in the syllogism, so I just proved that if an accidental predication were necessary then it would be essential. Then I went back and defined negation so permitting a direct denial that any accidental predication is necessary.

There are many theorems which one would naturally prove at this point, to facilitate further proofs and proof automation, which are not expressible syllogistically. Proof automation depends heavily on the demonstration of equations, so that proof may proceed by rewriting. But syllogisms are not suitable for this.

The natural way to proceed in such a case is to continue in this theory doing things which support proofs of syllogisms without being restrained to syllogisms, and then to have a separate theory in which the syllogistic claims are presented. Some reflection is desirable on what the philosophical objectives are and what course will best contribute to those purposes.

Here are some general modal results which I have not noticed in Aristotle as yet.

$\Vert\!\!\rangle\_elim\_thm =$
$\quad \vdash [\Vert\!\!\rangle\ P] \vDash P$
$\Diamond\_intro\_thm =$
$\quad \vdash [P] \vDash \Diamond\ P$
$\Vert\!\!\rangle\_\Diamond\_thm =$
$\quad \vdash [\Vert\!\!\rangle\ P] \vDash \Diamond\ P$

2.3.8 Other Conversions

The following conversions relate to the square of opposition, but I have not yet discovered where they appear in Aristotle. They work for all the copulas, so I have used a free variable for the copulas.

$$\begin{array}{l}
\vert \neg_All_conv_thm = \\
\vert \quad \vdash (\neg\ All\ A\ cop\ B) = Some\ A\ (cop\ not)\ B \\
\vert \neg_All_not_conv_thm2 = \\
\vert \quad \vdash (\neg\ All\ A\ (cop\ not)\ B) = Some\ A\ cop\ B \\
\vert \neg_Some_conv_thm = \\
\vert \quad \vdash (\neg\ Some\ A\ cop\ B) = All\ A\ (cop\ not)\ B \\
\vert \neg_Some_not_conv_thm = \\
\vert \quad \vdash (\neg\ Some\ A\ (cop\ not)\ B) = All\ A\ cop\ B
\end{array}$$

They are contraries out of Aristotles square of opposition

Normally theorems like this would be proved closed, but it looks more Aristotelian without the quantifiers and we can imagine that they are schemata. To use them it will usually be desirable to close them, which is easily done, e.g.:

SML

$$\vert all\_\forall\_intro\ \neg\_Some\_not\_conv\_thm;$$

ProofPower output

$$\vert val\ it = \vdash \forall\ A\ cop\ B\bullet\ (\neg\ Some\ A\ (cop\ not)\ B) = All\ A\ cop\ B : THM$$

2.3.9 Syllogisms for Essential Predication

Though the usual syllogisms are not valid for predication in general, the problems are confined to accidental predication. We can, by methods similar to those used above obtain automatic proofs of the 24 valid syllogisms restricted to essential predication.

The details are omitted, but the 24 izz syllogisms have been proven and stored in the theory, see: listing of theory *syllmetap*[**?**].

2.3.10 Some Accidental Syllogisms

2.3.11 Grice and Code

I now review the Grice/Code analysis under the revised interpretation of *izz* and *hazz*.

On my first attempt I did not notice till rather late that the material covered both Aristotle's and Plato's metaphysics, which are not wholly compatible. To do this using only conservative extension we have to use a different context for the parts of the treatment which might be incompatible. Three new theories will therefore be introduced, respectively covering:

gccom material common to Aristotle and Plato

gcaris material specific to Aristotle

gcplato material specific to Plato

All three theories are in the context of theory *syllmetap* which is an integrated model of both the modal syllogism and the metaphysics, incorporating the u-p syllogisms, which makes slightly more sense in the context of the metaphysics than otherwise.

Common Material

SML

```
force_new_theory "gccon";
```

The following results are now provable:

$$\begin{array}{ll}
\boldsymbol{FP1} = & \vdash [] \Vdash All\ A\ izz\ A \\
\boldsymbol{FP2} = & \vdash [All\ A\ izz\ B;\ All\ B\ izz\ C] \Vdash All\ A\ izz\ C \\
\boldsymbol{FP3} = & \vdash [All\ A\ hazz\ B] \Vdash \neg\ (All\ A\ izz\ B) \\
\boldsymbol{FP4a} = & \vdash [All\ A\ hazz\ B;\ All\ B\ izz\ C] \Vdash All\ A\ hazz\ C \\
\boldsymbol{FP4} = & \vdash [] \Vdash All\ A\ hazz\ B \Leftrightarrow (\exists_a\ C\bullet\ All\ A\ hazz\ C \wedge All\ C\ izz\ B)
\end{array}$$

These are not very Aristotelian. It would seem more Aristotelian to have:

$$\begin{array}{l}
\boldsymbol{FP3b} = \\
\quad \vdash [All\ A\ hazz\ B] \Vdash Some\ A\ (izz\ not)\ B
\end{array}$$

The above rendition of FP4 may not be true to the intention of Code. I possibly he might have intended that C be an individual.

To prove that more interesting result I need to define "individual".

HOL Constant

$$\boldsymbol{individual} : TermM \to BOOL$$

$$\forall A\bullet\ individual\ A \Leftrightarrow \exists b\bullet\ Snd\ A = NeSet\{b\}$$

Since the above is a regular predicate, we need something to convert an ordinary proposition into a modal proposition to use it in the context of modal syllogisms.

HOL Constant

$$\boldsymbol{Mp} : BOOL \to MPROP$$

$$\forall p\bullet\ Mp\ p = \lambda w\bullet\ p$$

The revised principle is then:

$$FP4b = ?\vdash [] \Vdash (All\ A\ hazz\ B \Leftrightarrow \exists\ C\bullet\ Mp(individual\ C) \wedge (All\ A\ hazz\ C) \wedge (All\ C\ izz\ B))$$

However, this is false in our model, since there need be no single attribute which is posessed by every substance which izz A. Consider the claim that all paints have colour. This may be true even if there is no individual colour which every paint hazz. However, if this stronger claim is not what Code intended, then what did he mean? Surely not the theorem I actually proved as FP4, since that is too trivial to be worth mentioning.

If we require A to be an individual we get a result:

$$\begin{array}{l} \boldsymbol{FP4c} = \\ \quad \vdash [Mp\ (individual\ A)] \\ \qquad \Vdash (All\ A\ hazz\ B \\ \qquad\quad \Leftrightarrow (\exists\ C\bullet\ Mp\ (individual\ C) \wedge All\ A\ hazz\ C \wedge All\ C\ izz\ B)) \end{array}$$

HOL Constant

$$\boldsymbol{Individual} : TermM \rightarrow MPROP$$

$$\forall A\bullet\ Individual\ A = \Vert\ \forall_a\ B\bullet\ All\ B\ izz\ A \Rightarrow_a All\ A\ izz\ B$$

Now on the face of it, in the context of our present model, the modal operator in this definition is irrelevant. This because all essential predication is necessary.

The following theorem confirms that intuition.

$$\begin{array}{l} individual\_lemma1 = \\ \qquad \vdash \forall\ A\bullet\ Individual\ A = (\forall\ B\bullet\ All\ B\ izz\ A \Rightarrow All\ A\ izz\ B) \end{array}$$

We can also show that the Code definition is equivalent to our own:

$$\begin{array}{l} individual\_lemma2 = \\ \qquad \vdash Individual\ A = Mp\ (individual\ A) \end{array}$$

In addition to the apparently spurious invocation of necessity, the Code definition depends upon the u-p syllogisms.

Our own primitive definition is couched in terms of the underlying model, and so in terms of that model we would have to regard as a primitive rather than a defined concept (this may be the best way to think of it).

Now we come to the Code definition of particular:

HOL Constant

$$\boldsymbol{particular} : TermM \to MPROP$$

$$\forall A\bullet\ particular\ A = \Downarrow \forall_a\ B\bullet\ All\ B\ are\ A \Rightarrow_a All\ A\ izz\ B \wedge_a All\ B\ izz\ A$$

A particular is an individual substance and one would have thought that a definition closer to saying that directly might have been a good idea. In this case in our model the modal operator is not redundant, because without it the definiens would be true if A were an individual attribute which contingently has an empty extension (i.e. is true of no substance), which is possible in this model. However, this cannot be true of necessity unless A is substantial.

Code might have used a similar device to define substantial:

HOL Constant

$$\boldsymbol{substantial} : TermM \to MPROP$$

$$\forall A\bullet\ substantial\ A = \Downarrow \forall_a\ B\bullet\ All\ B\ are\ A \Rightarrow_a All\ B\ izz\ A$$

and then defined a particular as a substantial individual. Alternatively substantial might be taken as primitive to avoid the use of a modal operator. Again, Code's definition relies on the u-p syllogisms.

Code's definition of universal is:

HOL Constant

$$\boldsymbol{universal} : TermM \to MPROP$$

$$\forall A\bullet\ universal\ A = \Diamond \exists_a\ B\bullet\ All\ B\ are\ A \wedge_a \neg_a\ (All\ A\ izz\ B \wedge_a All\ B\ izz\ A)$$

I think the intension is that a universal is anything except a particular, in which case that would be a better way to define it. However, in this model, this definition will be true of any non-individual, unless the $\ulcorner\Diamond\urcorner$ is changed to $\ulcorner\Downarrow\urcorner$.

Ontological Theorems T1-T3 are essentially redundant definitions so I will omit them and not use the additional vocabulary on this pass.

2.4 Conclusions

i	e_conv	conversion
ii	i_conv	
iii	ai_conv	
iv	eo_conv	
v	ae_obv	obversion
vi	ea_obv	
vii	io_obv	
viii	oi_obv	
ix	ao_contrad	contradiction
x	ei_contrad	
xi	ae_contrar	contrary
xii	io_subcontrar	subcontrary
xiii	ai_subalt	subalternate
xiv	eo_subalt	

Table 2.4: Laws of Immediate Inference

Chapter 3

Leibniz

This document has not really been started yet.

Here are some notes on what it might contain.

For the purposes of this analytic history the single most important concern is what Leibniz contributed to out understanding of the concept of logical truth. For this we consider primarily the most fundamental parts of his metaphysics, which we do partly through the perspective of Bertrand Russell [**?**], whose own philosophy of Logical Atomism was influenced by Leibniz and will be considered later.

Leibniz contributed also to our ideas about the applications of logic, through his "universal characteristic" and "calculus ratiocinator". Some kind of analysis of his ideas in this area would be nice.

3.1 Leibniz On Identity

This just shows the triviality of the identity of indiscernibles in our logical context, and raises the question, what more substantive point is Leibniz making and has it any substance?

It is clear that Leibniz's intention was not to forumlate such trivial principles as are found here. He intends that distinct individuals always differ in some substantive way (using that term informally). The real problem here is whether this can be captured formally in HOL, and this section at present does not make offer any elightenment on that topic.

SML

```
open_theory "misc2";
force_new_theory "leibniz01";
set_pc "misc2";
```

In higher order logic an identity of indiscernables (though probably not Leibniz's) is a trivial principle.

Its formulation is:

$$\forall x\ y\bullet (\forall P\bullet P(x) \Leftrightarrow P(y)) \Rightarrow x = y$$

Here is a long-winded transcript of a ProofPower proof session:

SML

```
set_goal([], ⌜∀x y• (∀P• P(x) ⇔ P(y)) ⇒ x = y⌝);
```

ProofPower output

```
(* *** Goal "" *** *)

(* ?⊢ *)  ⌜∀ x y• (∀ P• P x ⇔ P y) ⇒ x = y⌝
```

Strip the goal.

SML

```
a (REPEAT strip_tac);
```

ProofPower output

```
(* *** Goal "" *** *)

(*  1 *) ⌜∀ P• P x ⇔ P y⌝

(* ?⊢ *) ⌜x = y⌝
```

Instantiate the assumption using the predicate ⌜ $= y ⌝. [1]:

SML

```
a (spec_asm_tac ⌜∀ P• P x ⇔ P y⌝ ⌜$= y⌝);
```

ProofPower output

```
(* *** Goal "" *** *)

(*  2 *) ⌜∀ P• P x ⇔ P y⌝
(*  1 *) ⌜y = x⌝

(* ?⊢ *) ⌜x = y⌝
```

The instantiation yields:

$$
\begin{aligned}
&\ulcorner y = x \Leftrightarrow y = y \urcorner \\
&\Leftrightarrow \ulcorner (y = x \Rightarrow y = y) \wedge (y = y \Rightarrow y = x) \urcorner \\
&\Leftrightarrow \ulcorner (\neg y = x \vee y = y) \wedge (y = x \vee \neg y = y) \urcorner \\
&\Leftrightarrow \ulcorner (\neg y = x \vee T) \wedge (y = x \vee F) \urcorner \\
&\Leftrightarrow \ulcorner (y = x) \urcorner
\end{aligned}
$$

of which the last is the new assumption shown above.

Rewrite the conclusion with the assumptions (giving ⌜$x = x$⌝ which is automatically discharged).

SML

```
a (asm_rewrite_tac[]);
```

ProofPower output

```
Tactic produced 0 subgoals:
Current and main goal achieved
```

[1] This is the predicate "equal to y", or ⌜$\lambda x.y = x$⌝

Save the theorem.

SML

```
val leibniz_identity = save_pop_thm "leibniz_identity";
```

ProofPower output

```
Now 0 goals on the main goal stack
val leibniz_identity = ⊢ ∀ x y• (∀ P• P x ⇔ P y) ⇒ x = y : THM
```

So, in this context, that indiscernibles are identical is an elementary consequence of the fact that for every entity 'e' there is a predicate 'equal to e' which is satisfied only by e.

Leibniz intended by his principle something more substantial, which is harder to capture.

3.2 The Calculus Ratiocinator

This section is primarily based on what is said about Leibniz in the book written by Lukasievicz on Aristotles syllogistic, I have not checked this out against Leibniz's own writings, though it seems plausible from what I have read.

Leibniz's calculus is an arithmetisation of Aristotle's syllogistic. That such an arithmetisation will have the power which Leibniz attributes to it is certainly not the case, but my concern here is just to build a model which is similar to the arithmetic interpretation and allows us to check the extent to which it properly captures the relevant parts of Aristotle's logic. Since useful groundwork on this is done in my formal treatment of Aristotle[**?**], I will make use of some of that material by making this document logically dependent upon it, and making the theory which is here developed a child of one of the my models of Aristotle.

3.2.1 Leibniz's Interpretation of Aristotle's Syllogistic

The rationale for Leibniz's interpretation of propositions depends upon his conceptual atomism. This is the idea that concepts can be classified as either *simple* or *complex* and that complex concepts are defined ultimately (though possibly indirectly) in terms of simple concepts, by limited means. The limited means consist of negation of simple concepts and conjunction. It is further assumed that simple concepts are logically independent of each other, and that none of them is always true or always false (possibly this should be read necessarily true or necessarily false).

Given this simple idea of what concepts are, conceptual inclusion is decidable provided that we know which concepts are simple and we know the definitions of all the complex concepts. Conceptual inclusion corresponds to the A form of proposition, the I form is also decidable, and the other two are defined in terms of those two.

Leibniz arithmetises this by assuming that each simple concept is given a unique prime number, and that complex concepts are then represented by two numbers. The first of these two numbers is the

product of the primes of the simple concepts which occur positively in the definition of the complex concept (when this has been expanded out so that it no longer mentions any complex concepts and therefore consists of a conjunction of simple concepts or their negations). The second number is the product of the primes which code the simple concepts whose negations appear in the expanded definition.

Arithetisation is not essential, any equivalent way of coding up the information about which simple concepts or negations of simple concepts appear in the definition of a complex concept will do, and reasoning will be simpler if the problem of obtaining prime factorisations is sidestepped. We need not know how simple concepts are represented, and we can represent a conjunction as a list of conjuncts.

There is a small awkwardness if we want equality of concepts to be the same thing as equality of the representation and hence obtain:

> *A a B and B a A entails A = B*

If we just used two pairs of lists of simple concepts then the same concept would have multiple representatives, and pairs of lists which overlapped would not represent concepts at all. We could use a pair of sets of simple concepts, but then we have the possibility of infinite sets and we still might have overlap. A function from simple concepts into the type BOOL+ONE where the BOOL component represents negative or positive presence of the simple concept gets the identity correct but might allow infinite numbers of simple concepts. This is a possible point of divergence from Leibniz, but I'm going to try this one.

SML

```
open_theory "aristotle";
force_new_theory "leibniz02";
```

Semantics

We will allow that any type be used as the simple concepts, so that all the rules we can establish will be correct in the finite case and in the infinite case.

SML

```
declare_type_abbrev("TermL", [], ⌜:'a → TTV⌝);
```

TTV is a type with just three elements which may be thought of as truth values. Their names are *pTrue*, *pFalse* and *pU*. I will use *pTrue* to make a positively occurring simple concept, *pFalse* to mark a negated simple concept. *pU* marks a simple concept which does not occur in the relevant complex concept.

The predicate *Some* will be true iff a TTV does not have the value *pU*.

HOL Constant

$$\boldsymbol{Some} : TTV \to BOOL$$

$$\forall x \bullet Some\ x = \neg\ x = pU$$

The predicate $IsPos$ will be true iff a BOOL+ONE has the value $pTrue$.

HOL Constant

$$\boldsymbol{IsPos} : TTV \to BOOL$$

$$\forall x \bullet IsPos\ x \Leftrightarrow x = pTrue$$

Predication

"o" is already in use for functional composition, so we will use "u" instead and then use an alias to permit us to write this as "o" (type inference will usually resolve any ambiguity).

To render these in HOL we first declare the relevant letters as infix operators:

They predication operators are defined as follows:

SML

```
declare_infix (300, "a");
declare_infix (300, "e");
declare_infix (300, "i");
declare_infix (300, "u");
```

HOL Constant

$$\$\boldsymbol{a} : TermL \to TermL \to BOOL$$

$$\forall A\ B \bullet A\ a\ B \Leftrightarrow \forall x \bullet (B\ x = pTrue \Rightarrow A\ x = pTrue) \wedge (B\ x = pFalse \Rightarrow A\ x = pFalse)$$

HOL Constant

$$\$\boldsymbol{i} : TermL \to TermL \to BOOL$$

$$\forall A\ B \bullet A\ i\ B \Leftrightarrow \forall x \bullet A\ x = B\ x \Rightarrow A\ x = pU$$

HOL Constant

$$\$\boldsymbol{e} : TermL \to TermL \to BOOL$$

$$\forall A\ B \bullet A\ e\ B \Leftrightarrow \neg\ A\ i\ B$$

HOL Constant

$$\$\boldsymbol{u} : TermL \to TermL \to BOOL$$

$$\forall A\ B \bullet A\ u\ B \Leftrightarrow \neg\ A\ a\ B$$

SML

```
declare_alias("o", ⌜$u⌝);
```

Note that as defined above these come in complementary pairs, a being the negation of o and e of i. If we had negation we could manage with just two predication operators.

The Laws of Immediate Inference

Though in the source of this kind of "literate script" are to be found the scripts for generating and checking the proofs of all the theorems presente, it will not be my practice to expose these scripts in the printed version of the document. These scripts are not usually intelligible other than in that intimate man-machine dialogue which they mediate, and sufficient knowledge for most purposes of the structure of the proof will be found in the detailed lemmas proven (since the level of proof automation is modest).

However, I will begin by exposing some of the scripts used for obtaining proofs of syllogisms in this model, to give the reader an impession of the level of complexity and kind of obscurity involved in this kind of formal work, I will not attempt sufficient explanation to make these scripts intelligible, they are best understood in the interactive environment, all the scripts are available for readers who want to run them.

Most readers are expected to skip over the gory details, the philosophical points at stake do not depend on the details of the proofs.

Before addressing the laws of immediate inference [2] I devise a tactic for automating simple proofs in this domain.

The following elementary tactic expands the goal by applying the definitions of the operators and then invokes a general tactic for the predicate calculus. A rule is also defined using that tactic for direct rather than interactive proof.

[2] in which I followed Strawson [?], though I can now cite Aristotle, Prior Analytic, Book 1, Part 2. [?]

SML

```
val syll_tacL = REPEAT (POP_ASM_T ante_tac)
      THEN rewrite_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝])
      THEN REPEAT strip_tac THEN all_asm_fc_tac []
      THEN_TRY asm_rewrite_tac[];
fun syll_ruleL g = tac_proof (g, syll_tacL);
val syll_tacLb = REPEAT (POP_ASM_T ante_tac)
      THEN rewrite_tac (map get_spec [⌜$a⌝, ⌜$e⌝, ⌜$i⌝, ⌜$u⌝])
      THEN contr_tac THEN asm_fc_tac[];
fun syll_ruleLb g = tac_proof (g, syll_tacLb);
```

```
val e_conv_thm = A e B ⊢ B e A : THM
val i_conv_thm = A i B ⊢ B i A : THM
```

Simple Conversion

Conversion Per Accidens These are OK.

```
val ai_conv_thm = A a B ⊢ B i A : THM
val eo_conv_thm = A e B ⊢ B o A : THM
```

Obversion For these we need to define an operation of complementation on terms.

HOL Constant

```
Complement : TermL → TermL
```

```
∀A α• (Complement A) α =
          if A α = pTrue then pFalse
          else if A α = pFalse then pTrue
          else pU
```

We will use "~" as a shorthand for "Complement".

SML

```
declare_alias ("∼", ⌜Complement⌝);
```

Only two of the obversions are valid.

```
val ae_obv_thm = A a B ⊢ A e ∼ B : THM
val iu_obv_thm = A i B ⊢ A o ∼ B : THM
```

The Square of Opposition

$$
\begin{array}{l}
|\ ao\_contrad\_thm = \vdash\ A\ a\ B \Leftrightarrow \neg\ A\ o\ B \\
|\ ei\_contrad\_thm = \vdash\ A\ e\ B \Leftrightarrow \neg\ A\ i\ B \\
|\ ae\_contrar\_thm = \vdash\ \neg\ (A\ a\ B \wedge A\ e\ B) \\
|\ io\_subcont\_thm = \vdash\ A\ i\ B \vee A\ o\ B \\
|\ ai\_subalt\_thm = \vdash\ A\ a\ B \Rightarrow A\ i\ B \\
|\ eo\_subalt\_thm = \vdash\ A\ e\ B \Rightarrow A\ o\ B
\end{array}
$$

The Syllogisms

First we make a *mapkit*.

SML

$$
|\ val\ sLmapkit{:}mapkit = mkSimpMapkit\ \ulcorner{:}TermL\urcorner\ [\ulcorner\$a\urcorner,\ulcorner\$e\urcorner,\ulcorner\$i\urcorner,\ulcorner\$u\urcorner];
$$

Then we apply this in generating and proving the syllogisms.

SML

$$
\begin{array}{l}
|\ proveGoals\ syll\_tacL\ \texttt{""}\ (mkGoals\ sLmapkit\ syllogism\_data1); \\
|\ proveGoals\ syll\_tacL\ \texttt{""}\ (mkGoals\ sLmapkit\ syllogism\_data2); \\
|\ proveGoals\ syll\_tacL\ \texttt{""}\ (mkGoals\ sLmapkit\ syllogism\_data3);
\end{array}
$$

The theorems are also displayed in the theory listing in listing of theory *leibniz02*[**?**]

3.3 Metaphysics

This is an adaptation of the model of Aristotelian logic and metaphysics in section 2.3, to reflect the most crucial differences between Aristotle and Leibniz.

Russell [**?**] represents Leibniz as having adhered rather strictly to Aristotle's logic with bad consequences for his metaphysics, in particular he sees the idea of monads as having arisen from the idea that all propositions have subject/predicate form. This is something into which I hope to look more closely in due course.

However, our analysis of Aristole suggests that if his metaphysics is so construed as to make his logic sound, then existence is necessary, and this view is incorporated into our model (which may go too far in this matter). For Leibniz however the position on existence is pretty clear. Existence of substantial individuals is the only thing which is contingent, all else is necessary.

The following model is therefore an exploration of what happens if we tweak the underlying model to ensure exactly that. One thing that we should expect, is that the syllogisms which exhibit the "existential fallacy" will no longer be sound. The second thing which seems to flow from that is

the irrelevance of the essential/accidental distinction which is possibly the most important feature of Aristotelian metaphysics. In 2.3 I assume that this distinction is intended to correspond to that between necessary and contingent truth (though this may be tendentious, I am not aware of explicit textual support for it). This can no longer be done, and I therefore abandon Aristotelian metaphysics altogether returning to a treatment of the syllogism less influenced by metaphysics.

The connection with Grice and Codd is no longer relevant so that material also is excised.

SML

```
open_theory "misc2";
force_new_theory "leibniz03";
force_new_pc "'leibniz03";
```

3.3.1 The Subject Matter

Once the essential/accidental distinction is discarded, we are left with a metaphysic in which the key distinction is between individual substances and predicates.

We take an individual to be something which is only truly predicable of itself, and other predicates as collections of individuals, once again accounting for (essential) predication as set inclusion (having represented an individual as a unit set). This subset relation is fixed, but the individual substances which are the extensions of predicates are themselves contingent.

SML

```
new_type ("ISUB", 0);
```

Let us take a new type for a fixed population of predicates.

SML

```
new_type ("PRED", 0);
```

Whose extension is fixed.

HOL Constant

$$\boldsymbol{extension} : PRED \to ISUB\ \mathbb{P}$$

$$T$$

However, the extension is defined in terms of individual substances whose existence is contingent, and so we still have the possibility of distinguishing essential predication from accidental, according to whether inclusion obtains on the full extension, or merely on the extensions restricted to the individuals which actually exist.

A possible world is therefore a collection of individuals.

SML

```
declare_type_abbrev ("W", [], ⌜:ISUB ℙ⌝);
```

We to distinguish one possible world which is the actual world:

HOL Constant

```
actual_world : W
```

```
T
```

Subjects and predicates are just things of type *PRED*.

3.3.2 Predication

Though the retreat from accidental predication simplifies matters I will retain a presentation of the syllogism similar to that in Section 2.3, for the sake of its readability.

So I separate out the quantifier by defining *All* and *Some* appropriately, and retain the postfix negator even though only one kind of predication is now available. (in fact I could define the two kinds of predication because the I still have available two kinds of extension, but the modal operators suffice to express the distinction between the two kinds of predication).

I will then treat the modal operators as operators over propositions, and introduce the syllogism as a kind of judgement.

The type of the primitive copulas is:

SML

```
declare_type_abbrev("COPULA", [], ⌜:ISUB → PRED → (W → BOOL)⌝);
```

The first parameter is an individual substance rather than a PRED, the quantifying operato will arrange for each of the relevant individuals to be supplied.

SML

```
declare_type_abbrev ("MPROP", [], ⌜:W → BOOL⌝);
```

Propositions

Complementation The distinction between affirmative and negative is achieved by a postfix negation so we can say “is not”, or “are not” (which in this models would be synonyms, so we will go with “are” only.

SML

$declare\_postfix\ (100,\ "not")$;

HOL Constant

$\$\boldsymbol{not} : COPULA \to COPULA$

$\forall pred\bullet\ pred\ not = \lambda pa\ t\ w\bullet \neg\ pred\ pa\ t\ w$

Quantifiers The quantifiers expect to be supplied with a copula and a term. The quantifier then predicates using the copula the term of everything or something in the domain of quantification (which is the subject term). The copulas are defined below.

HOL Constant

$\boldsymbol{All} : PRED \to (ISUB \to PRED \to MPROP) \to PRED \to MPROP$

$\forall\ s\ r\ p\bullet\ All\ s\ r\ p = \lambda w\bullet\ \forall z\bullet\ z \in extension\ s \wedge z \in w \Rightarrow r\ z\ p\ w$

HOL Constant

$\boldsymbol{Some} : PRED \to (ISUB \to PRED \to MPROP) \to PRED \to MPROP$

$\forall\ s\ r\ p\bullet\ Some\ s\ r\ p = \lambda w\bullet\ \exists z\bullet\ z \in extension\ s \wedge z \in w \wedge r\ z\ p\ w$

Predication For essential predication it is necessary that the individual and the predicate are both of the same category and then reduces under our model to set membership. In effect. since the non-substantial individuals are tagged with their category, we need only deal separately with the distinction between substantial and non-substantial and the set inclusion will ensure a match in the non-substantial categories.

HOL Constant

$\boldsymbol{are} : ISUB \to PRED \to MPROP$

$\forall\ i\ t\bullet\ are\ i\ t = \lambda w\bullet\ i \in extension\ t$

Modal Operators In this model the model operators are operators over propositions.

HOL Constant

$\Diamond : MPROP \to MPROP$

$\forall p\bullet\ \Diamond\ p = \lambda w\bullet\ \exists w'\bullet\ p\ w'$

HOL Constant

$$\Big| \; \rrbracket : MPROP \to MPROP$$

$$\Big| \; \forall p\bullet \rrbracket\ p = \lambda w\bullet \forall w'\bullet\ p\ w'$$

3.3.3 Propositional Operators

Though the truth functional propositional operators do not feature in the syllogism it is nevertheless useful to have them in giving a full account of Aristotle's logic and they are therefore here defined.

That these propositional operators are "truth functional", in a context in which propositions are not regarded as denoting truth values requires a little explanation perhaps. Our propositions are families of truth values indexed by possible worlds, i.e. functions from possible worlds to truth values, or in the context of a two valued logic (which Aristotle's seems to be), sets of possible worlds. In this context the usual truth functional operators can be expressed by mapping the usual operator over the set of possible worlds, i.e. the result in every possible world is the result of applying the truth functional operator to the values of the propositions in that possible world. These also correspond to the obvious set theoretic operation if the propositions are thought of as sets of possible worlds, i.e. intersetion for conjunction, complementation for negation.

The symbols for the operators are already in use, so we define the operations using decorated variants of the symbols and use an alias to allow the undecorated symbol to be used.

HOL Constant

$$\Big| \; \neg_{\boldsymbol{a}} : MPROP \to MPROP$$

$$\Big| \; \forall p\bullet \neg_a\ p = \lambda w\bullet \neg\ (p\ w)$$

SML

$$\Big| \; declare\_alias\ (\texttt{"}\neg\texttt{"},\ \ulcorner\neg_a\urcorner);$$

SML

$$\Big| \; declare\_infix(220,\ \texttt{"}\wedge_a\texttt{"});$$

HOL Constant

$$\Big| \; \$\wedge_{\boldsymbol{a}} : MPROP \to MPROP \to MPROP$$

$$\Big| \; \forall p\ q\bullet (p \wedge_a q) = \lambda w\bullet (p\ w) \wedge (q\ w)$$

SML
```
declare_alias ("∧", ⌜$∧a⌝);
```

SML
```
declare_infix(210, "⇒a");
```

HOL Constant
```
$⇒a : MPROP → MPROP → MPROP
```

```
∀p q• (p ⇒a q) = λw• p w ⇒ q w
```

SML
```
declare_alias ("⇒", ⌜$⇒a⌝);
```

SML
```
declare_infix(200, "⇔a");
```

HOL Constant
```
$⇔a : MPROP → MPROP → MPROP
```

```
∀p q• (p ⇔a q) = λw• p w ⇔ q w
```

SML
```
declare_alias ("⇔", ⌜$⇔a⌝);
```

3.3.4 Quantification

The Grice/Code analysis makes use of quantifiers, particularly existential quantification. To verify the formulae in this context we therefore need to define modal version of the quantifiers.

SML
```
declare_binder "∀a";
```

HOL Constant
```
$∀a : (PRED → MPROP) → MPROP
```

```
∀mpf• $∀a mpf = λw• ∀t• mpf t w
```

SML

```
declare_alias ("∀", ⌜$∀a⌝);
```

SML

```
declare_binder "∃a";
```

HOL Constant

```
$∃a : (PRED → MPROP) → MPROP
```

```
∀mpf• $∃a mpf = λw• ∃t• mpf t w
```

SML

```
declare_alias ("∃", ⌜$∃a⌝);
```

3.3.5 Judgements

I'm not yet clear what to offer here, so for the present I will define two kinds of sequent, which will be displayed with the symbols ⊨ asnd ⊩ . the former being a kind of contingent material implication and the latter a necessary implication.

Both form of judgement seem suitable for expressing the rules of the syllogism at first glance but which can also be used for conversions.

The first expresses a contingent entailment, that if some arbitrary finite (possibly empty) collection of premises are contingently true, then some conclusion will also be true. Since the consequence is material, and the premisses might be contingent, the conclusion might also be contingent. One might hope that if the rules of the syllogism are applied and the premises are necessary, then so will be the conclusions.

SML

```
declare_infix(100, "⊨");
```

HOL Constant

```
$⊨ : MPROP LIST → MPROP → BOOL
```

```
∀lp c• lp ⊨ c ⇔ Fold (λp t• p actual_world ∧ t) lp T ⇒ c actual_world
```

This one says that in every possible world the premises entail the conclusion (still material).

SML

```
declare_infix(100, "⊩");
```

HOL Constant

$\$\Vdash : MPROP\ LIST \to MPROP \to BOOL$

$\forall lp\ c\bullet\ lp \Vdash c \Leftrightarrow \forall w\bullet\ Fold\ (\lambda p\ t\bullet\ p\ w \wedge t)\ lp\ T \Rightarrow c\ w$

In the present context the choice between the two is probably immaterial, since we know no more about the actual world than any other, so anything that we can prove to be true contingently, we can also prove to be true necessarily.

3.3.6 Conversions

Premisses, their Modes and Conversions See: Prior Analytics Book 1 Part 2 Paragraph 2.

> First then take a universal negative with the terms A and B.
>
> If no B is A, neither can any A be B. For if some A (say C) were B, it would not be true that no B is A; for C is a B.
>
> But if every B is A then some A is B. For if no A were B, then no B could be A. But we assumed that every B is A.
>
> Similarly too, if the premiss is particular. For if some B is A, then some of the As must be B. For if none were, then no B would be A. But if some B is not A, there is no necessity that some of the As should not be B; e.g. let B stand for animal and A for man. Not every animal is a man; but every man is an animal.

These work out fine for *izz*, so I will do those first, and then show that they fail for *hazz* and *is*.

The first and third conversions are most useful when expressed as an equation, since our proof system is based primarily on rewriting using equations.

are_not_lemma =

$\vdash All\ B\ (are\ not)\ A = All\ A\ (are\ not)\ B$

some_are_lemma =

$\vdash Some\ B\ are\ A = Some\ A\ are\ B$

These we also supply as our Aristotelian judgements, together with the second which does not give an equation. The second conversion embodies the existential fallacy, and therefore is not provable here.

$$\boldsymbol{are\_conv1} = \vdash$$
$$[All\ B\ (are\ not)\ A] \Vdash All\ A\ (are\ not)\ B$$

$$\boldsymbol{are\_conv2} = \vdash$$
$$[All\ B\ izz\ A] \Vdash Some\ A\ izz\ B$$

$$\boldsymbol{are\_conv3} = \vdash$$
$$[Some\ B\ izz\ A] \Vdash Some\ A\ izz\ B$$

3.3.7 Modal Conversions

Prior Analytics Book 1 Part 3 See: Universal and Possible Premisses and their Conversions.

These are the conversions in relation to necessity and possibility described by Aristotle:

1. If it is necessary that no B is A, it is necessary also that no A is B.
2. If all or some B is A of necessity, it is necessary also that some A is B.
3. If it is possible that all or some B is A, it will be possible that some A is B.
4. and so on

So in this section Aristotle only offers variants of the previous conversions with either "possible" or "necessary" attached to both premiss and conclusion.

We can prove generally that modal operators can be introduced into a conversion:

$$\boldsymbol{\Diamond\_conv} =$$
$$\vdash [P] \Vdash Q \Rightarrow [\Diamond\ P] \Vdash \Diamond\ Q$$

$$\boldsymbol{\rangle\!\rangle\_conv} =$$
$$\vdash [P] \Vdash Q \Rightarrow [\rangle\!\rangle\ P] \Vdash \rangle\!\rangle\ Q$$

Now according to Leibniz all predication is necessary, only existence is contingent. However, the contingency of existence means that this must be interpreted as a claim about predicates applied only to individuals.

$$\boldsymbol{\Diamond AllBareA\_thm} = \vdash [] \Vdash \Diamond\ (All\ B\ are\ A)$$

The upshot is that to show that our model captures the necessity of predication (in the sense in which this is conceivable), we need a way to talk about individuals.

HOL Constant

$$\boldsymbol{individual} : PRED \to MPROP$$

$$\forall A\bullet\ individual\ A = \lambda w\bullet\ \exists a\bullet\ extension\ A = \{a\}$$

$$\boldsymbol{\Diamond AarenotA\_thm} = \vdash [] \Vdash \Diamond\ (All\ A\ (are\ not)\ A)$$

There are many theorems which one would naturally prove at this point, to facilitate further proofs and proof automation, which are not expressible syllogistically. Proof automation depends heavily on the demonstration of equations, so that proof may proceed by rewriting. But syllogisms are not suitable for this.

The natural way to proceed in such a case is to continue in this theory doing things which support proofs of syllogisms without being restrained to syllogisms, and then to have a separate theory in which the syllogistic claims are presented. Some reflection is desirable on what the philosophical objectives are and what course will best contribute to those purposes.

Here are some general modal results which I have not noticed in Aristotle as yet.

$$\boldsymbol{⦇\_elim\_thm} =$$
$$\vdash [⦇\ P] \vDash P$$
$$\boldsymbol{\Diamond\_intro\_thm} =$$
$$\vdash [P] \vDash \Diamond\ P$$
$$\boldsymbol{⦇\_\Diamond\_thm} =$$
$$\vdash [⦇\ P] \vDash \Diamond\ P$$

3.3.8 Other Conversions

The following coversions relate to the square of opposition, but I have not yet discovered where they appear in Aristotle. They work for all the copulas, so I have used a free variable for the copulas.

$$\neg\_All\_conv\_thm =$$
$$\vdash (\neg\ All\ A\ cop\ B) = Some\ A\ (cop\ not)\ B$$
$$\neg\_All\_not\_conv\_thm2 =$$
$$\vdash (\neg\ All\ A\ (cop\ not)\ B) = Some\ A\ cop\ B$$
$$\neg\_Some\_conv\_thm =$$
$$\vdash (\neg\ Some\ A\ cop\ B) = All\ A\ (cop\ not)\ B$$
$$\neg\_Some\_not\_conv\_thm =$$
$$\vdash (\neg\ Some\ A\ (cop\ not)\ B) = All\ A\ cop\ B$$

They are contraries out of Aristotles square of opposition

Normally theorems like this would be proved closed, but it looks less Aristotelian without the quantifiers and we can imagine they are schemata. To use them it will usually be desirable to close them, which is easily done, e.g.:

SML

```
all_∀_intro ¬_Some_not_conv_thm;
```

ProofPower output

```
val it = ⊢ ∀ A cop B• (¬ Some A (cop not) B) = All A cop B : THM
```

3.3.9 Syllogisms

The abolition of accidental predication should result in a syllogistic logic which corresponds to Aristotle, though the contingency of existence means that the existential fallacies really are fallacies.

We can, by methods similar to those used above obtain automatic proofs of the syllogisms which are valid in this model.

The details are omitted, but the valid syllogisms have been proven and stored in the theory, see: listing of theory *leibniz03*[**?**].

Chapter 4

Hume and Kant

Chapter 5

Frege

Chapter 6

Russell and Wittgenstein

6.1 Introduction

This is barely started and isn't worth reading yet!

6.1.1 Preliminary Ideas

The following aspects of these works are those which first strike me as significant for this history.

1. Russell on Leibniz (this is to appear in [**?**] but there might be connections with material here).
2. Russell on Denoting (connection with Frege[**?**] on sinn and bedeutung and Grice on vacuous names [**?**]).
3. Russell's theory of types
4. Russell on incomplete symbols and logical fictions
5. Russell's logical atomism
6. Wittgenstein's Tractatus (connection with Carnap).

More generally and vaguely:

1. the conceptions of analytic method
2. the conception of logical truth and of The Fork.
3. semantics, truth functional, truth conditional.
4. metaphysics

Logical Atomism

From Russell's introductory remarks one can easily get the impression that Russell is giving a presentation of some of Wittgenstein's ideas. In the main, however, Russell's logical atomism is an account of his theory of logical types, which predates any contrubution from Wittgenstein. Wittgenstein's contribution (to Russell's conception of Logical Atomism) seems primarily semantic/metaphysical insight. I put these together since it seems to me that what comes across as metaphysics is better construed non-metaphysically as semantics, this constituting a softening of the doctrines. One might say, that the question is addressed "what must the world be like in order for it to be spoken of in this way", and the step from there to model teory is not so great. Of course Wittgenstein would not have liked it, just as he rejected Russell's characterisation of his Tractatus as concerning idealised languages.

The Tractatus, truth functions and conditions

Wittgenstein's Tractatus seems to me, in those small parts which I have so far found to be intelligible, to centre around the two ideas, firstly that propositions express truth functions of atomic propositions, and secondly that logical truths are tautologies. Wittgenstein's treatment of this yields only a narrow conception of logical truth, easily seen to be equivalent to first order validity. But he has in mind analyticity. And a full conception of analyticity can be seen to fall under this same idea, the idea of something which allways evaluates to true. Wittgenstein's treatment is defective, there are huge holes, the whole of mathematics and the examples, such as colour exclusion, of atomic propositions which are not logically independent (or at least, not analytically dependent).

In the period immediately after the completion by Whitehead and Russell of Principia Mathematica there were attempts to draw out from this achievement some philosophical lessons. The most conspicuous figure in this is Wittgenstein and his contribution was the Tractacus Logico-Philosophicus [**?**]. Russell, appearing to follow in his wake (though prior to the publication of the Tractatus) gave lectures on Logical Atomism [**?**, **?**]. [1]

6.2 Notes on Russell

This section contains some rough notes on relevant parts of Russell's writings.

6.2.1 Principia Mathematica

Preface

Note that in their Preface Whitehead and Russell point out that:

> The explanation of the hierarchy of types given in the Introduction differs slightly from that given in *12 of the body of the work. The latter explanation is stricter and is that which is assumed throughout the rest of the book.

He does not go into further detail at this point, but I believe the difference consists in the omission in the Introduction of mention of the ramifications to the type system.

[1] Two other relevant documents of a more technical nature are worth mentioning here, to get a sense of how the technical content of these philosophical works relates to that appearing in the emergent discipline of mathematical logic. These are [**?**, **?**]. These connect with specific features of the Tractatus and Logical Atomism respectively.

Introduction to First Edition

Ch. II. The Theory of Logical Types

This is a representation of the ideas first presented in [**?**].

Russell opens by recognising that the theory of types was adopted because it appears to resolve the known paradoxes, but claims nevertheless that the theory is consonent with "common sense" and is "inherently credible" for which reason it is presented first in its own right before its credentials in resolving the paradoxes are discussed.

I. The Vicious Circle Principle Vicious circles of the kind Russell here considers, arise from the supposition that some collection of objects may have members "which can only be defined by means of the collection as a whole". Such a collection is to be regarded (possibly following Cantor) as an illegitimate totality, and any proposition involving such a totality is meaningless.

In the following discussion it appears that Russell supposes that objects under consideration are created by their definition, and that this can only be accomplished if all the entities referred to (even through quantification) in the definition exist prior to the object being defined.

The vicious circle principle is then stated:

> "Whatever involves *all* of a collection must not be one of the collection."

Already by the time that *Principia Mathematica* is published Russell has found this principle to be too heavy handed, and has found it necessary to mitigate its effects by adopting the axiom of reducibility.

It is worth discussing briefly how this proposal looks from a contemporary perspective, and then noting when some of these contemporary insights first appear.

First consider some distinctions which Russell does not appear to note:

i. that between defining some totality and defining some element of it.

ii. between various kinds of involvement, e.g. between *containing* and *being defined in terms of*.

iii. that between circularity and vicious circularity.

Thus, one might suppose that if some totality is already definite, to pick out an element of the totality by means of reference to the whole does not involve circularity and should not be problematic. If we have some account of an ontology which does not depend upon some particular method of defining elements of the totality, we may then introduce a notation in which definitions may be expressed in a manner which presumes that the totality itself is already in place and can be quantified over. In this case however, there can be no presumption that formulation of a definition brings into being or in itself establishes the existence of the thing defined, there is a need to establish either individually or for

general kinds of definition that the objects defined do exist in the totality in question. If we define an object by definite description, then we must be assured that some single thing satisfies that description in the already defined domain of discourse, if by indefinite description, that at least one thing satisfies the description.

Even if the totality in question is already definite, circularity in a definition may be a problem, but in many cases will not be. Given such a definite totality the acceptability of circularity in definitions is a matter which is resolved by a proof of the existence in the totality of an element which conforms to (satisfies) the definition.

Russell's principle will seem to many heavy handed (perhaps not to constructivists), and though Russell has to moderate the effects of the principle by adopting the axiom of reducibility, he does not offer or justify a moderation of the principle itself. The effect of Russell's axiom of reducibility is generally held to be to make his system equivalent to a simple rather than a ramified type theory. The simple type theory can be understood as achieving its effect not through prohibition of circularity of definitions (as understood by Russell, i.e. as disallowing quantifiction over the type of the object defined) but by ensuring the well-foundedness of the underlying ontology. Well-foundedness prevents an object involving the whole of a totality of which it is a member, in another sense, in the sense that the transitive closure of the membership relation may not be reflexive.

[More and better discussion is needed here.]

II. The Nature of Propositional Functions Russell begins the discussion of propositional function as if he did not have a notation for functional abstraction, and as if a function were written in exactly the same way as we would write the body of a functional abstraction. His discussion sounds as if he has confused the function with the body of such a functional abstraction for quite some time, until he finally draws the distinction, and introduces his notation for functional abstraction. This work of course precedes the advent of the lambda-calculus so the terminology I used was not available to him, and my understanding of the status of variable binding constructs is not well understood by me. Russell was previously acquainted with Frege Grundlagen in which there is notation for conceptual abstraction.

Russell defines a propositional function as a proposition which contains a free (*real* in his terminology) variable. Russell has a notation for class abstraction, and a variety of other variable binding constructs such as universal and existential quantifiers and definite description, but at this point he enters into the discussion of functions as if he had no notation for functional abstraction.

He talks at first of propositional functions as we would today talk about formulae containing free variables. This he does at the outset when he identifies the *essential characteristic* of propositional functions as *ambiguity*. This is connected with the fact that a propositional function (as he now speaks of it) can be asserted, and that such an assertion is to be understood as asserting something about an undetermined individual.

A function is only determinate if each of its values is determinate, which they will be only if (though not necessarily if) they do not involve the function. This is an application of the vicious circle principle. The function presupposes its values, but is not presupposed by them. Thus "Socrates is mortal" is presupposed by buy does not presuppose the propositional function "x is mortal". The latter might

possibly involve vicious circularity even though some of its instances do not.

Now Russell introduces his notation for functional abstraction [2], which consists in putting a circumflex over the variable to be bound, in the course of making the distinction between the function itself and an indeterminate value of the function. Nevertheless he continues to regard the function itself, rather than the formula with the free variable, as ambiguously denoting one of its instances. Of course, we would now take the functional abstraction as denoting the function, and the body before the abstraction possibly as denoting an ambiguous instance (though we explain this more precisely by allowing that it denotes only in a context in which the value of the variable has been determined).

Because of the vicious circle principle, we will have to say that for some values of its argument a propositional function has no value (for example, the argument cannot be the propositional function itself. The range of significance of the function consists of the collection of values for which the function is significant.

III. Definition and Systematic Ambiguity of Truth and Falsehood Truth and Falsehood as predicates must be considered confined to a single type of argument for the sake of compliance with the vicious circle principle as is supposed necessary for the avoidance of contradictions. Their informal use should therefore be construed as exemplifying systematic ambiguity and as asserting implicitly that particular instance necessary for the argument to which they are applied.

IV. Why a Given Function requires Arguments of a Certain Type Russell explains this in terms of the essence of functions, which he has given as ambiguity. By this he means what we would today describe as the fact that a function must be supplied with an argument.

The use of a variable function symbol in the body of an expression determines the type which an argument to the function defined by that body must have.

V. The Hierarchy of Functions and Propositions From the above considerations we find ourselves with a hierarchy of functions of ascending type. The individuals, functions over individuals, functions over functions of individuals, and so on.

This picture must be further complicated because the functions over individuals do not together constitute a legitimate totality, and must themselves be divided into a hierarchy.

Russell's description of what is now called his ramified type system is very sketchy, particularly in the introductions where you might expect it to be spelled out fully[3], it is necessary to look into the body of the work for further enlightenment, which might even then leave room for doubt. Combined with the lack of explicit type annotations arising from the use of systematic ambiguity, this makes it not straightforward to extract a precise account of the ramified type system and leaves doubt for any particular proposal as to whether it is what Whitehead and Russell intended. The secondary literature is sometimes also imprecise or inaccurate in presenting what Whitehead and Russell do say.

[2] Chapter II section II, p40 second paragraph

[3] But note that he says in his preface that the fuller account is to be found in *12

Whitehead and Russell are most explicit in their description of how the order of a propositional function is determined. However, it is clear that the order is a coarser classification than the types. The information which Russell supplies about types consists only in the general principle that a type is the range of significance of a propositional function, and in various additional observations suggesting when types are distinct.

Russell's talk about functions is exclusively about propositional functions, which may have multiple arguments, in which case they are relations. Propositional functions are only of the same type if they take the same number of arguments and the types of the arguments are the same. There are no functions which deliver values other than propositions, but there are individuals.

It is clear that propositions (not just propositional functions), and individuals, do not constitute a totality and are not all of the same order. In these cases the order is determined by the maximum order of the bound variables which occur in their definition. A reservation is present in relation to individuals, since a definition of an individual which involves quantifiers must be a description, which is an incomplete symbol, and the possibility arises that this might negate the rationale for giving an order to an individual. Be that as it may, a propositional function which takes an individual as an argument must have a type distinct from one which takes a proposition in the same place and this leads us to the following account of Russell's ramified type heirarchy.

A type is:

1. The type of Propositions of order n (for any natural number n¿0)
2. The type of Individuals of order n (in which there remains some doubt about whether n may be other than 1)
3. A type of propositional functions of order n over a finite sequence of arguments of specified types (where n must be greater than the order of every argument).

Hylton's Possible Misreading The following issue I first came across when reading Peter Hylton's excellent volume on Russell's early work [**?**]. At first I went along with Hylton since the passage he cited seemed unambiguous, but further rummaging lead me to doubt that Russell had meant what Hylton (and I) took him to mean. First I describe the issue as it appeared to me on first reading Hylton. Then I explain my grounds for believing Hylton to have misunderstood Russell.

The above account of the type distinctions envisaged by Russell seems to be incomplete. A discussion of the additional conditions may be found in [**?**] p305. Russell's remarks suggest that more information about the type of bound variables in the definition should influence the type of the definiens[4]. To incorporate this into the system would complicate all three clauses in the above definition, since the types of both propositions and individuals might then have to include the types of bound variables occuring in the definition, and the type of propositional functions would also have to include this

[4]'First order propositions are not all of the same type, since, as was explained in *9, two propositions which do not contain the same number of apparent variables cannot be of the same type.' Principia Mathematica [**?**] *12, p162.

information. In each case this would either involve a set or a multi-set of types of apparent variables (but possibly only the multiplicity).

This last elabortion, though evidently thought desirable by Russell, is difficult to motivate. Without it, the type system is well motivated first by the need to ensure that arguments to propositional functions must match in number and kind the use of the real variables in the definition otherwise nonsense will ensue, and by the dictates of the vicious circle principle which is incorporated through the concept of order. Having ensured compliance with the vicious circle principle by incorporation of order in to the type, further information about the types of apparent variables in the definition seems to serve no purpose, and if included potentially makes the information about order redundant (the order of a propositional function is obtained by adding one to the maximum order of the real and apparent variables in its definition).

On the matter of whether Russell really intended the number of quantifiers in a definition to affect its type, I have followed through the text more carefully and come to doubt that he did. There is of course reason to doubt it because he does not offer any reason for such an elaboration. The passage cited by Hylton ([**?**] *12, p162) itself refers elsewhere (*9) for the explanation, and does not explicitly speak of quantifiers. It speaks of *apparent variables* which include those bound by quantifiers, but also the variables for the arguments of a propositional function (even when they are not circumflexed). There is reason to believe that he intends here *only* the variables for arguments, not quantified variables.

When we look back to *9 to discover why Russell might think the *number* of quantifiers relevant, there does not appear to be any explanation. What we do find ([**?**] *9, p128) is a discussion of how the propositional variables in sections *1-*5 'are necessary elementary propositions' and that the primitive propositions must be restated for first order propositions and analogues of the propositions in *2-*5 obtained by 'merely repeating the same proofs'. It follows he says that the process can be repeated to obtain a similar theory for propositions with two apparent variables so on indefinitely. By this process he says, 'propositions of any order can be reached'. This is a little odd, for there are two distinct directions of movement here, adding an extra quantified variable of the same type would not by his own account change the order of a proposition, though one can of course progress through the orders by progressively increasing the order of even a single quantified variable in the propositional. What does change the type, but not necessarily the order, is the movement from propositions to propositional functions, and that of adding additional apparent variables as arguments rather than for quantification. Section *9 does not therefore supply a rationale for considering the number of apparent variables relevant to the order of a proposition, or of a propositional function, or even any further grounds for our belief that Russell held the number to be relevant. However, here and later in sections *10 and *11 (on the theory of one and two apparent variables respectively), we see that the subject matter does the relevant numbers of quantified variables, and also that number of argument places in propositional functions, and offers nothing to confirm the idea that type distinctions arise merely from the presence of greater numbers of quantifiers.

Fuller Treatment of the Ramified Type System A full account of the ramified type system, by today's standards would also involve a more formal statement of the above definition, a semantics assigning domains to the types, and an account of well-typed formulae of the theory of types. I may

make some steps in this direction in the formal materials below, but this is of lesser interest than other aspects of Russell's philosophy and may therefore not happen. (of course, this has already been done many times)

Predicativity The term predicative has had many uses since it was introduced by Russell, apparently in 1906. Feferman provides an extended account in [**?**] in [**?**].

Apparently Russell at first used the term for propositional functions which defines a class (one whose extension exists), by contrast with the kinds of function whose extension is incoherent, which he called impredicative. The general idea that predicativity priginally meant no more than consistency is worth bearing in mind, for this is potentially a much more liberal notion of predicativity than those which have succeeded it.

In Principia Russell uses the term in a quite different way, which is important because of the role it plays in the formulation of the axiom or reducibility. A propositional function is predicative if its order is no greater than is required by the types of its arguments, i.e. if it does not quantify over any type of higher order than the highest of its arguments.

6.2.2 Part I

Section A The Theory of Deduction

Section B Theory of Apparent Variables

***9 Extension of the Theory of Deduction from Lower to Higher Types of Propositions** Here Russell talks about typical ambiguity and justifies its use.

He also provides more information about the type system, and it is here that he refers to (from *12) for an explanation of his talk about the effect of the multiplicity of apparent variables on the type of a propositional function (in *12).

***12 The Hierarchy of Types and the Axiom of Reducibility**

6.3 Preliminary Observations

At the core of both Russell's atomism and Wittgenstein is the metaphysical idea that the world consists of a collection of facts. [5]

This is probably one of the ideas Russell took from Wittgenstein. Though this features in both systems, its has a different role in each. Wittgenstein's exploitation of this idea is more thorough

[5] I need to find out where this idea first appeared.

and systematic than Russell's. In Wittgenstein the fundamental insight is that propositions express truth functions of atomic propositions, and that logical truths (which Wittgenstein seems to identify with analytic propositions) are those which are tautologous (i.e. true whatever the truth values of the atomic propositions, i.e. whatever are the atomic facts). Together with the insistence that the atomic facts are logically independent, we have a faulty account of analyticity, but a good account of the semantics of first order logic. Without this general insistence on logical independence of atomics, we have a conception of semantics which is important and influential.

Russell builds his logical atomism around the same core, atomic facts. But its clear firstly that the simple idea that propositions are truth functional does not inspire him, and that he has important objectives which are absent in Wittgenstein. The pure logical semantics we find commuted in simplistic way into metaphysics by Wittgenstein tells us little or nothing about what the things are which are related by atomic propositions.

6.3.1 Schematic Variables

It is explicit in PM, under the heading of "typical ambiguity", that the propositions therein are to be regarded as schemas. This works by omission of type information from variables, and it intension is that a formula can be read with any assignment of types which is type correct (though that idea is not precisely defined).

It is also the case that Russell's use of "incomplete symbols" does involve the use of certain symbolic expressions in ways which can only be understood by syntactic elimination of the symbol together with some of the syntactic context in which they occur, The best known example of the use of incomplete symbols arises from Russell's theory of desriptions, described in his classic paper *On Denoting*. Even more significant perhaps in PM is the use of incomplete symbols to effect Russell's 'no classes' treatment of set theoretic terminology.

It has been further suggested that PM makes use of *schematic variables*, and in particular when names such as f appear unbound they are not to be regarded as free functional variables varying over propositional functions of some type, but as schematic variable varying of syntactic expressions. I have so far seen no convincing evidence for this point of view, which is seems to me is contradicted (though not quite explicitly) by what Russell says about variables on page of PM.

The evidence I have seen adduced is that certain propositions in *20 only make sense with such a schematic interpretation rather than as containing real functional variables.

I probe this a little here using ProofPower. Of course, this tool does not implement the logic of PM, it implements a polymorphic version of Church's Simple Type Theory. However, for the issues at hand I think this may be close enough. The propositions in question concern the elimination of set theoretic vocabulary. *20.07 defines the application of a propositional variable to class, and must be regarded as expressing a syntactic transformation. The definiendum involves a symbol for a class. Since classes do not exist, this cannot be an object language variable, all of which range over individuals or propositional functions. It therefore seems that the definition shows a stage in the elimination of class terminology in favour of terminology not involving classes. It is the application here which is being

eliminated. However, there seems no reason to regard the propositional function being applied on the right as schematic.

When expressed in ProofPower the point of *20.07 is largely lost. What is says is, that when some function is asserted of all classes (of a type), that should be read as asserting it of all predicative functions (over that type).

SML

```
open_theory "rbjmisc";
force_new_theory "PM01";
set_pc "rbjmisc1";

declare_type_abbrev("CLASS", ["'a"], ⌜:'a → BOOL⌝);

val PM20p07 = ⌜(∀α:IND CLASS•f(α)) = ∀ϕ:IND → BOOL• f(ϕ)⌝;
```

6.4 The Tractatus

I will begin by attempting to present a formal model of (aspects of) Wittgenstein's Tractatus[**?**].

SML

```
open_theory "rbjmisc";
force_new_theory "tract01";
set_pc "rbjmisc1";
```

6.4.1 The World

Wittgenstein says that a world consists of facts rather than things, that it is all the facts, that these tell us both what is the case and what is not the case, that they are "in logical space" and that they are logically independent.

I take this to be analogous to atomic propositions, the requirement that we know both what is and is not the case is satisfied if we have all the true atomic proposition, without any negated propositions, and this also best meets the requirement for logical independence. We do not know here what a fact is so the information we have here is best caputured by a type constructor or a function (depending on whether we think the facts will be a type or a set).

I will assume that it will be a type (this can probably be arranged). WIttgenstein, has implicitly alluded to possible worlds in 1.21 (though is not necessarily 'ontologically committed'), and we might think of the type ⌜:('a)WORLD⌝ being the type of all possibile worlds whose facts have type ⌜:'a⌝.

SML
```
declare_type_abbrev("WORLD", ["'a"], ⌜:'a ℙ⌝);
```

The world is then, something of type ⌜:('a)WORLD⌝, which I will refrain from introducing until we know more about what a fact is.

6.4.2 States of Affairs

A state of affairs is some combination of objects.
SML
```
new_type("OBJECT", 0);
```

It must therefore involve a collection of objects possibly with some information about how they are combined (allowing that the same set of objects might be combined in more that one way).

Because we later find other kinds of object which have similar structure to states of affairs, we will use the same type constructor in each case, applying the constructor to different types as appropriate.

In order to give structure to a collection we must first distinguish the elements of the collection and then supply some recipe for
SML
```
declare_type_abbrev("CONSTRUCTION", ["'a", "'b", "'c"], ⌜:('a → 'b + ONE) ×'c⌝);
```

In the above definition, a *CONSTRUCTION*, is a type constructor of which the first type parameter is a type of indexes or tags, the the second a type of things. This represents collections of objects which are indexed rather than an unstructured, allowing the same object to be used more than once, and allowing that an object be supplied for inclusion in the structure in some specific place. The third type parameter is the type of the information indicating how the constuents are to be combined to form the whole.

6.4.3 Thoughts and Propositions

The things which stand for objects in propositions are names.
SML
```
new_type("NAME", 0);
```

6.4.4 Propositions as Truth Functions - I

Wittgenstein introduces a single truth functional connective, which is a generalised distributed denial (generalised Scheffer stroke). [6]

[6] The non-generalised version was discovered by Scheffer, the generalised version was presented by Schonfinkel in his paper on combinatory logic (presented in 1920, published in 1924 [**?**, **?**]).

This is readily defined in HOL. We define it as an operator on sets of functions, since that allow us to use set displays as the arguments.

HOL Constant

N: $(BOOL)\mathbb{P} \to BOOL$

$\forall s\bullet\ N\ s \Leftrightarrow \forall p\bullet\ p \in s \Rightarrow \neg\ p$

We can then prove that all the usual logical connectives are definable in terms of ⌜N⌝. The following theorems show that the logical constants in HOL are equivalent to expressions in N.

$N\_\neg\_thm = \vdash\ \forall\ p\bullet\ (\neg\ p) = N\ \{p\}$
$N\_\wedge\_thm = \vdash\ \forall\ p\ q\bullet\ (p \wedge q) = N\ \{N\ \{p\};\ N\ \{q\}\}$

HOL Constant

M: $('a \to BOOL) \to BOOL$

$\forall f\bullet\ M\ f \Leftrightarrow \forall x\bullet\ \neg\ f\ x$

$M\_\neg\_thm = \vdash\ \forall\ p\bullet\ (\neg\ p) = M\ (\lambda\ x\bullet\ p)$
$M\_\wedge\_thm =$
$\quad \vdash\ \forall\ p\ q\bullet\ (p \wedge q) = M\ (\lambda\ x\bullet\ \mathit{if}\ x\ \mathit{then}\ M\ (\lambda\ x\bullet\ p)\ \mathit{else}\ M\ (\lambda\ x\bullet\ q))$
$M\_\forall\_thm = \vdash\ \forall\ pf\bullet\ \$\forall\ pf = M\ (\lambda\ x\bullet\ \neg\ pf\ x)$
$M\_\exists\_thm = \vdash\ \forall\ pf\bullet\ \$\exists\ pf = M\ (\lambda\ x\bullet\ M\ (\lambda\ x\bullet\ pf\ x))$

6.4.5 Propositions

HOL Constant

N_2: $(BOOL)\mathbb{P} \to BOOL$

$\forall s\bullet\ N\ s \Leftrightarrow \forall p\bullet\ p \in s \Rightarrow \neg\ p$

Chapter 7

Grice

I will endeavour to follow the structure of Grice's paper *Vacuous names*[**?**, **?**]. However, the method employed for the formal analysis, which is called "shallow embedding", is essentially semantic, and involves addressing semantic fundamentals before finding manageable syntactic presentations, and I therefore find it necessary to address key aspects of the semantics before considering the details of syntax.

7.0.6 Background

Grice's deliberations about Vacuous Names may be understood primarily against those who have supposed or desired a significant difference between formal logics and the logic of ordinary discourse, by one who felt that the differences are, or need only be, modest. In some of his writings. for example in the "Retrospective Epilogue" in *Sudies in the Way of Words*[**?**], Grice describes two camps, "Modernists" and "Neo-traditionalists", the former making more and the latter less of the distinctions between formal and ordinary logic.

Elsewhere in Grice's philosophy, in connection with this controversy, the relationship between the logical connectives and their ordinary language counterparts is discussed. Here we are concerned with the problem of referring expressions which lack a referent.

Modern discussion of this problem begins with Russell [**?**, **?**], who sought formal languages for mathematics which were more precise and more transparent in their logical structure than is ordinary language.

There are two aspects of the problems here at stake which are separable. One concerns logic and language; how we talk and reason using words or phrases which do not, or might not, refer to any existent entity. The second is the metaphysical question of what exists. Russell's most important work in this area arose from his making a transition away from a lavish Meinongian ontology in which there are different kinds or grades of existence, to the adoption of Occam's razor, leading to a spartan ontology. Key to this is the acceptance that some referring phrases fail to refer, and Russell's principal contribution here is in his theory of descriptions [**?**, **?**].

Russell's theory treats definite and indefinite descriptions as "incomplete symbols", which contribute to the meaning of the sentence as a whole even if they fail to refer. Such incomplete symbols are only meaningful in certain particular context, and their meaning is given by translated the whole into some other phrase in which the incomplete symbol does not occur and in which there remains no explicit reference to an entity which might not exist. Usually this involves quantification. Proper names are then eliminated in favour of descriptions.

Grice's principal aim is to resurrect the conception of names as references, rather than as surrogates for descriptions. He wants to do this without being forced to regard names of non-existent entities, such as Pegasus, as yielding sentences which are neither true nor false. This he achieves by allowing that names which do not designate nevertheless refer to some kind of entity which he calls its correlate, and by the rule that predications to non-designating names are to yield falsehoods.

7.0.7 Preliminary Formalities

I consider two different approaches, both derived from Grice, to the problems discussed in *Vacuous Names*[**?**]. Much of the discussion concerns matters which are common to both, and this involves some formal material. In order to avoid replication the formalisation of this common ground is provided first in a separate theory before matters specific to the two variants are entered into.

Common material is in the theory 'grice' whose listing may be found in appendix **??**.

```
SML
open_theory "rbjmisc";
force_new_theory "grice";
```

7.1 The Problem

Grice uses several lists in his presentation. To simplify unambiguous reference I have prefixed some of the numbering schemes with capital letters.

Our starting point is a list of eight *inclinations* supplied by Grice and summarised here:

I1 That individual constants be admitted.

I2 Note that names are sometimes "vacuous"

I3 Thence that a constant might lack a designatum.

I4 That excluded middle and bi-valence nevertheless remain unqualified.

I5 That a claim about (predication to) a non-designating constant be false and its negation true.

I6 That no unusual constraints on "U.I." [1] and "E.G." [2] be introduced.

I7 That the law of identity (in the form $\forall x \bullet x = x$) be a theorem[3].

I8 That derivability implies entailment.[4]

In relation to these Grice points out two *difficulties* as follows:

(a) I2, I3 and I7 between them seem *prima facie* to enable proofs of the existence of non-existent entities (e.g. "Pegasus exists").

[1] Universal Instantiation
[2] Existential Generalisation
[3] Which seems to suggest that identity is not a predicate.
[4] i.e. that the system be sound!

(b) I5 and I6 would if gratified permit us to infer that there exists something which does not fly from the premise that Pegasus does not exist.

He then lists five possible *resolutions* the difficulties:

R1 To resist I3 and insist that constants have a designatum.

R2 Resisting I4 and I5, insist that predications to non designating terms and their negations lack a truth value.

R3 Resist I1 and do without individual constants.

R4 Resist I2 by insisting that all constants have a designatum, which might perhaps "be" without existing.

R5 Resist I6 by adding requirements on existential generalisation that the relevant constant be shown to designate.

R6 Resist I8, allowing that the deductive system be not strictly sound, but only so subject to the condition (or "marginal" assumption) that all names have bearers.

Grice does not discuss any of these alternatives, but instead proposes to attempt to square the circle by devising a system which satisfies all these (apparently) incompatible inclinations.

7.2 System Q: Objectives

Grice now proposes a first order predicate calculus meeting two particular objectives, spelt out in five points of further detail.

(i) That a sentence such as "Pegasus does not fly" be capable of rendition in two distinct ways one of which will be true and the other false in the case that Pegasus does not exist. These correspond to disambiguations of the logical structure of the sentence either as predicating non-flying or denying a predication as flying.

(ii) That in either case the inference from "Pegasus does not fly" to "there is something which does not fly" is to be admitted.

O1 U.I. and E.G. are acceptable without special side conditions.

O2 Some sentences involving non-denoting constants will be true and provable.

O3 It will be formally decidable whether a sentence depends for its truth on whether some constant designates.

O4 It will be possible to find in Q representation of sentences such as "Pegasus exists".

O5 There will be an extension of Q in which identity is represented.[5]

7.3 Classical Logic using HOL

Grice tries to make his system as close as possible to "classical logic", basing his system on a first order predicate calculus presented as a natural deduction system based on a textbook by Mates [**?**]. I hope that the pertinent sense of "classical" here is *two-valued*.

In this presentation we work with a "classical" higher order logic (based on Church's formulation of the Simple Theory of Types[**?**]) in the form of a sequent calculus. In this section we replicate in this system theorems which show the closeness of the relationship between this classical logic and the one from which Grice departs.

Grice discusses the problem of the scope of names before getting into the details of his classical logic, but since we here adopt a different manner of resolving this problem which depends upon special features of our "classical logic", the discussion of scope belongs here.

7.3.1 Scope

A part of the novelty and difficulty in system Q lies in its special provisions for controlling the "scope" of names.

The use of the word scope in this context is distinct from its more common usage (at least in mathematical logic and computer science) in which the (or a) scope of a name (variable or constant) is a syntactic region within which it may be used with the same sense (as opposed to uses of the same name for entirely distinct purposes as may appear in different scopes of the name).

In this case the relevant ambiguity of scope is not a region of significance of the name, but the extent of some predicate applied to the name. Thus in the example used by Grice, there is an ambiguity in the sentence "Pegasus does not fly" about the logical structure of the sentence which leaves doubt about what predicate is being applied to Pegasus. We may regard the sentences as predicating "x does not fly" of Pegasus, or we may be denying that the predicate "flies" applies to Pegasus. The question is whether the negation is part of the predicate or is applied to the result of the predication.

In System Q Grice provides a predicate logic in which numerical subscripts may be used to disambiguate the predications which are taking place. Unfortunately, this notation, like the dot notation in Principia Mathematica is not as readable as the more usual use of brackets for disambiguating scope. Furthermore, this aspect of the syntax would be hard to replicate by the methods we propose to adopt, which are intended to permit a lightweight analysis of semantic issues and their connection with the validity of sentences and the soundness of derivations.

[5] No discussion here of whether identity is a predicate.

In our analysis we will therefore provide for the desired control over predication by the use of Church's lambda notation. We expect that our syntax will be readily related to that of System Q, but since the system we formalise is at least syntactically distinct we will call it system C[6].

System C is obtained by conservative extension to HOL, which is a version of higher order logic derived directly from Church's Simple Theory of Types [**?**]. Church's STT is distinctive for its very slender logical core, achieved by building the logical system from the simply typed lambda calculus. In this logical system boolean valued functions serve for predication, and the ideas of Grice on vacuous names are realised principally by defining a different kind of predication. To minimise confusion, the word predicate will only be used when Grice's notion of predication is intended, otherwise propositional or boolean valued function will be used.

The use of lambda notation for making explicit the predicate to be applied may be illustrated simply using Grice's Pegasus example. In using this method we must distinguish two kinds of predication. The kind whose scope we are now discussing, which is the only kind of predication in System Q, and the kind of predication which we have in HOL, consisting in the application of a propositional function to some argument. Henceforth these two will be distinguished by using the term "predication" only for the former, by making this kind of predication explicit in the concrete syntax as a copula, and by using the phrase "application of a propositional function" for the second kind of predication. The use of a copula not only visually distinguishes Griceian predication, but also allows us to control its semantics through the definition of the copula.

The copula will be the word "is", and since we will be discussion more than one variant of Grice's system, the copulas will be distinguished by subscripts giving the name of the system to which they belong. In System C, then, the copula is "is_c", so the application of the predicate "*Flier*" to the term "*Pegasus*" will be written "*Pegasus* is_c *Flier*".

The two variants of "Pegasus does not fly" are then expressed as:

"$\neg$ (*Pegasus* is_c *Flier*) "

and

"*Pegasus* is_c ($\lambda x \bullet \neg$ *Flier* x)".

In the first case the predicate applied to Pegasus is $\ulcorner Flier \urcorner$, in the second case it is $\ulcorner \lambda x \bullet \neg Flier\ x \urcorner$ (which is a negation of Flier, "not a flier").

This technique makes it possible to examine the semantic aspects of System Q using formal machine assisted deductive reasoning, while avoiding the special intricacies of formalising Grice's special syntax. Whether this pays off in terms of perspicuity and penetration in the analysis remains to be seen.

We will therefore offer no further discussion of Grices notational devices for scoping at this point, but may consider later how much may have been lost by this approach.

At this stage I am looking at two alternatives for the underpining semantics, and the following two

[6]This in homage to Rudolf Carnap, Q having been chosen by Grice in homage to Quine, and G having been chosen by Myro[**?**, **?**] in homage to Grice.

section work through these two alternatives. In each case, we are devising a "shallow embedding" of something close to Grice's System Q in ProofPower HOL. These embedded aproximations to System Q are called System C (in homage to Rudolf Carnap) and System S (a nod to Speranza).

7.3.2 Semantic Domains

Grice goes on from his discussions of scope to the presentation of the remaining aspects of the syntax, notably of the various aspects of the deductive system.

We will track this discussion of inference in a semantic way, demonstrating the semantic principles which ensure the soundness of the deductive system presented in a manner which makes reasonably clear the relationship between the two. This amounts to a semantic embedding into HOL of a system which is intended to be semantically parallel to system Q, though not syntactically quite the same.

To do this we must first provide some basic definitions on which the semantics is based, and before doing that I must explain part of the method of semantic embedding which might seem to, but does not in fact, prejudice one of the principle issues at stake. That is that the embedding consists in a development of a semantics which is purely denotational, i.e. in which every expression has a denotatum. These denotation need not be what one would naturally suppose the reference of an expression to be, they may be purely abstract surrogates which suffice to describe the truth conditions of the language. So the way we model a language in which term expressions may not denote, is by taking the range of things which they might possibly denote, and adding one or more extra items which are values which they are give to indicate that they have no denotation.

Interpretations

Grice's definition of an interpretation of Q (VIII A of [**?**]) is ambiguous in what it says about the domain of an interpretation. The domain is a set of correlates, some of which may be unit sets the element of which is a designatum. Grice tells us that there need not be any such unit sets, but does not say explicitly that there may be no other correlates. He does tell us that he has in mind that there will be exactly one non-designating correlate, the empty set, but he does not place this as a requirement, so the significance of his having this in mind is unclear.

There are three possibilities concerning the number of non-designating correlates in an interpretation, that there are none, exactly one, or more than one. In relation to these there are a number of possible positions in relation to the definition of an interpretation, the most obvious are:

1. no constraint
2. there must be at least one
3. there must be exactly one

Of course there may be more, but I propose to discuss just these three.

From a literal reading of Grice in this section I would take the first. However, from the fact that he explicitly allows there to be no designata but does not explicitly say this for non-designating correlates, we might infer that he intends but does not state that there will be at least one non-designating correlate. Alternatively we might take the case he had in mind more presciptively and work with item 3.

These are significant semantically, of course.

Allowing interpretations with no designata will create problems with descriptions if these were to be introduced into Q, for one would then have no correlate available for a non-designating descriptions. It also affects a matter which Grice later discusses, which is whether certain existential claims are valid, notably the claim that there is something which does not exist. This (or something similar "$\exists x_4.\ \neg_3\ F_1\ x_2$", VIII B p137) Grice later claims to be valid, and this is evidence against interpretation 1 in favour of 2 or 3.

Item 2, though the one perhaps preferred by Grice, has the disadvantage that all non-designating terms have the same correlate, and will therefore be equal in a sense of equality which satisfies the universal law of reflexivity of identity. Do we want Pegasus and Sherlock Holmes to be identical?

Nevertheless, in our shallow embedding of Q we will adopt 2, for, in addition to its possible preferment by Grice, and its confirming Grice's belief in the validity of the cited existentials, it allows a simpler and more transparent shallow embedding than either of the others. *[fuller explanation to be supplied]*

First let us give a name for the domain of discourse of an interpretation of the language Q. We will suppose this to be a "type" in HOL, but will leave open the type by using a type variable. To serve as denotions for terms in system G_{HP} we use a disjoint union of two types, the first of which is the type of those things which do exist (candidate designata) and the second is a type of surrogates ("correllates" in Grice) for those things which do not exist. We use V for Value.

7.3.3 Descriptions

When Grice comes to discuss descriptions he has in mind a distinction between descriptions and names which would impact upon the kind of interpretation necessary for a semantic embedding. This in turn would impact the detailed definition of the copula.

To make apparent the differences which would arise in treating descriptions as Grice intends we provide two treatments of those aspects of the logical system which depend upon the details of the semantics. These will be called System C and System S.

Certain aspects of our presentation, particularly those which simply illustrate the correspondence between the classical propositional logic in System Q and that in ProofPower-HOL, are common to System C and System S and are therefore treated first.

7.3.4 Propositional Logic

The following material is common to Systems C and S and shows how these systems relate to the propositional logic in System Q. Since we are working with a semantic embedding rather than a syntactic meta-theory we cannot prove results about derivations and instead present proofs about the semantics which show that the required inference rules would be sound.

To do this we adopt a notation for expressing entailment in C in a manner parallel to the presentation of the natural deduction system of Q. This is done by defining the infix relationship ($al \models c$) holding between a list of assumptions (or premises) al and a single conclusion c when the conclusion is entailed by the premises.

SML

```
declare_infix(4, "⊨");
```

HOL Constant

$$\$\models : BOOL\ LIST \to BOOL \to BOOL$$

$$\forall al\ c\bullet\ (al \models c) \Leftrightarrow (\forall_L\ al) \Rightarrow c$$

In the following theorems which capture the content of Grices rules for Q as theorems of C, the variables ϕ, ψ, ζ... range over formulae (boolean terms) and Γ, Δ, Θ... over lists (finite sequences) of formulae. P ranges over propositional functions, values whose types are instances of $\ulcorner$:'a $\to$ BOOL$\urcorner$ (in which $'a$ is a type variable). The notation "$[\phi; \psi; ...]$" is an explicit list formation ("$[\phi]$" being a single element list) and the symbol "$\frown$" is the concatenation operator over lists.

In general the rules of Q can be formulated as theorems of HOL because in a higher order logic, and can quantify over predicates. This means that where substitution is involved in a rule, this is captured as a theorem by the use of a variable as a predicate applied to a variable as an argument. When such a theorem is instantiated using a lambda expression as the predicate, the substitution into the expression will be achieved by beta-reduction.

$$\begin{aligned}
\boldsymbol{ASS} &= \vdash \forall\ \phi\bullet\ [\phi] \models \phi \\
\boldsymbol{RAA} &= \vdash \forall \phi\ \psi\ \zeta\ \Gamma\bullet\ ([\phi] \frown \Gamma \models \psi \wedge \neg\psi) \Rightarrow (\Gamma \models \neg\phi) \\
\boldsymbol{DN} &= \vdash \forall\ \phi\bullet\ [\neg\ \neg\ \phi] \models \phi \\
\wedge\boldsymbol{I} &= \vdash \forall\ \phi\ \psi\bullet\ [\phi; \psi] \models \phi \wedge \psi \\
\wedge\boldsymbol{E} &= \vdash \forall\ \phi\ \psi\bullet\ ([\phi \wedge \psi] \models \phi) \wedge ([\phi \wedge \psi] \models \psi) \\
\vee\boldsymbol{I} &= \vdash \forall\ \phi\ \psi\bullet\ ([\phi] \models \phi \vee \psi) \wedge ([\phi] \models \psi \vee \phi) \\
\vee\boldsymbol{E} &= \vdash \forall\ \phi\ \psi\ \zeta\ \Gamma\ \Delta\ \Theta\bullet\ ([\phi] \frown \Gamma \models \zeta) \wedge ([\psi] \frown \Delta \models \zeta) \wedge (\Theta \models \phi \vee \psi) \\
&\qquad \Rightarrow (\Gamma \frown \Delta \frown \Theta \models \zeta) \\
\boldsymbol{CP} &= \vdash \forall\ \phi\ \psi\ \Gamma\bullet\ ([\phi] \frown \Gamma \models \psi) \Rightarrow (\Gamma \models \phi \Rightarrow \psi) \\
\boldsymbol{MPP} &= \vdash \forall\ \phi\ \psi\bullet\ [\phi \Rightarrow \psi; \phi] \models \psi
\end{aligned}$$

7.3.5 Quantifier Inference-Rules

We now show the correspondence between the quantification rule in System Q with the standard quantifiers in HOL. Because of this correspondence we can use the standard HOL quantifiers in System C and get a good correspondence with System Q in this area. When we come to System S it will be necessary to redefine the quantifiers to obtain those qualifications on $\forall E$ and $\exists I$ which are necessary to cope with Grice's ideas about failing descriptions.

$$\begin{aligned}
&\forall \boldsymbol{I} = && \vdash \forall\ \Gamma\ P\bullet\ (\forall\ x\bullet\ \Gamma \vDash P\ x) \Rightarrow (\Gamma \vDash (\forall\ x\bullet\ P\ x))\\
&\forall \boldsymbol{E} = && \vdash \forall\ \Gamma\ P\ c\bullet\ (\Gamma \vDash (\forall\ x\bullet\ P\ x)) \Rightarrow (\Gamma \vDash P\ c)\\
&\exists \boldsymbol{I} = && \vdash \forall\ \Gamma\ P\ x\bullet\ (\Gamma \vDash P\ x) \Rightarrow (\Gamma \vDash (\exists\ x\bullet\ P\ x))\\
&\exists \boldsymbol{E} = && \vdash \forall\ P\ Q\bullet\ (\forall\ x\bullet\ [P\ x] \vDash Q) \Rightarrow ([\exists\ x\bullet\ P\ x] \vDash Q)
\end{aligned}$$

7.3.6 P Committal

The notion of E-committal is defined syntactically by Grice, and tells us which formulae in System Q are true only if some name has a designatum, i.e. to the existence of which named individuals our assertion of the formula would commit us.

In HOL we can formalise this idea without resort to reasoning about syntax, furthermore, the main detail in the definition is independent of whether we are in System C or System S, and is therefore provided here. To achieve this we parameterise the definition of E-committal so that it can be specialised to the different ways of asserting existence in Systems C and S, the parameterised version being called *P_committal*.

The formal criteria for *E_committal* then becomes provability in HOL (which is not in general decidable).

In Grice *E_committal* is a relationship, here it is a parameterised property of propositional functions (the parameter being the relevant notion of existence).

SML

```
declare_infix(100, "committal");
```

HOL Constant

$$\$\boldsymbol{committal} : ('a \to BOOL) \to ('a \to BOOL) \to BOOL$$

$$\forall P\ \phi\bullet\ P\ committal\ \phi \Leftrightarrow \forall\alpha\bullet\ \phi\ \alpha \Rightarrow P\ \alpha$$

The following theorems correspond to clauses in Grice's definition of *E_committal*:

$$\begin{array}{|ll}
\boldsymbol{P_2} = & \vdash \forall\ P\ \phi\bullet\ P\ committal\ \phi \\
& \quad \Rightarrow P\ committal\ (\lambda\alpha\bullet\ \neg\ \neg\ \phi\ \alpha) \\
\boldsymbol{P_3} = & \vdash \forall\ P\ \phi\ \psi\bullet\ P\ committal\ \phi \vee P\ committal\ \psi \\
& \quad \Rightarrow P\ committal\ (\lambda\ \alpha\bullet\ \phi\ \alpha \wedge \psi\ \alpha) \\
\boldsymbol{P_4} = & \vdash \forall\ P\ \phi\ \psi\bullet\ P\ committal\ \phi \wedge P\ committal\ \psi \\
& \quad \Rightarrow P\ committal\ (\lambda\ \alpha\bullet\ \phi\ \alpha \vee \psi\ \alpha) \\
\boldsymbol{P_5} = & \vdash \forall\ P\ \phi\ \psi\bullet\ P\ committal\ (\lambda\ \alpha\bullet\ \neg\ \phi\ \alpha) \wedge P\ committal\ \psi \\
& \quad \Rightarrow P\ committal\ (\lambda\ \alpha\bullet\ \phi\ \alpha \Rightarrow \psi\ \alpha) \\
\boldsymbol{P_6\forall} = & \vdash \forall\ P\ \phi\bullet\ (\exists\ \beta\bullet\ P\ committal\ (\phi\ \beta)) \\
& \quad \Rightarrow P\ committal\ (\lambda\ \alpha\bullet\ \forall\ \beta\bullet\ \phi\ \beta\ \alpha)
\end{array}$$

I have a problem proving the existential element of (6), so this clause is in doubt. I think Grice should be asking for "$\forall\beta \bullet\ \psi(\beta/\omega)$" to be *E_committal* for α (in system Q), which corresponds in systems C and S to the theorem:

$$\begin{array}{|ll}
\boldsymbol{P_6\exists b} = & \vdash \forall\ P\ \phi\bullet\ (\forall\ \beta\bullet\ P\ committal\ (\phi\ \beta)) \\
& \quad \Rightarrow P\ committal\ (\lambda\ \alpha\bullet\ \exists\ \beta\bullet\ \phi\ \beta\ \alpha)
\end{array}$$

(with P suitably instantiated).

7.4 System C

A new theory is needed which I will call "griceC" which is created here:

SML

```
open_theory "grice";
force_new_theory "griceC";
set_pc "rbjmisc";
```

Grice introduces a special notion of interpretation in terms of which to give the semantics of Q. An interpretation has a domain which is a set of correlates, some of which are unit sets, in which case the member of that unit set is called a designatum. For uniformity of type we assume that all correlates are sets of sets.

We allow the domain of an interpretation to be any type and for some element of that type to be identified as THE non-designating element.

HOL Constant

```
⊥: 'a
```

```
T
```

We can then use inequality with $\perp$ as the property of being a designatum. The structure of these interpretations is isomorphic to that of Grice's preferred notion of interpretation, even though we do not use unit sets for correlates of designating constants.

We may use HOL constants of appropriate type for Q constants and predicates, and HOL variables for Q variables, and the consequence should be that the theorems provable in HOL should be just those which are valid in Q.

This is done by using T and F instead of Corr(1) and Corr(0) respectively.

Now we define our special kind of predication. This is done by introducing 'is' subscripted with 'c' as a copula.

SML

$declare\_infix\ (400,\ "is_c");$

HOL Constant

$\$is_c\colon 'a \to ('a \to BOOL) \to BOOL$

$\forall\ p\ t\bullet\ t\ is_c\ p \Leftrightarrow if\ t = \perp\ then\ F\ else\ p\ t$

The definition says, if the term denotes then apply the predicate to the value denoted, otherwise the result of the predication is F (false).

Note that this will not work for relations, we would have to define a separate similar predicator for each arity of relation in use.

Here is the copula for applying a 2-ary relation.

SML

$declare\_infix\ (400,\ "is_{c2}");$

HOL Constant

$\$is_{c2}\colon ('a \times 'b) \to ('a \times 'b \to BOOL) \to BOOL$

$\forall\ p\ t\bullet\ t\ is_{c2}\ p \Leftrightarrow if\ Fst\ t = \perp \vee Snd\ t = \perp\ then\ F\ else\ p\ t$

Now let's introduce Pegasus. The "definition" just says that Pegasus is the non-denoting value.

HOL Constant

$\boldsymbol{Pegasus} : 'a$

$Pegasus = \perp$

And the predicate *Flier*. Note that the predicates apply only to real desigata under the normal HOL application, its only under our special predication that they become applicatble to possibly non denoting values.

HOL Constant

$$\boldsymbol{Flier} : 'a \to BOOL$$

$$T$$

We can now prove in HOL the elementary facts we know about Pegasus flying, yielding the following theorems:

$$\boldsymbol{Pegasus\_lemma\_1} =$$
$$\vdash Pegasus\ is_c\ Flier \Leftrightarrow F$$

$$\boldsymbol{Pegasus\_lemma\_2} =$$
$$\vdash Pegasus\ is_c\ (\lambda\ x\bullet \neg\ Flier\ x) \Leftrightarrow F$$

$$\boldsymbol{Pegasus\_lemma\_3} =$$
$$\vdash \neg\ Pegasus\ is_c\ Flier$$

$$\boldsymbol{Pegasus\_lemma\_4} =$$
$$\vdash \neg\ Pegasus\ is_c\ (\lambda\ x\bullet \neg\ Flier\ x)$$

In case you may doubt the neutrality of our use of lambda expressions, we show here that they are inessential to this approach, and can be dispensed with in favour of explict definitions of the predicates required. Thus we may instead define a predicate which is the denial of *Flier*:

HOL Constant

$$\boldsymbol{NonFlier} : 'a \to BOOL$$

$$\forall x\bullet\ NonFlier\ x \Leftrightarrow \neg\ Flier\ x$$

We can then prove the slightly reworded:

$$\boldsymbol{Pegasus\_lemma\_5} = \vdash Pegasus\ is_c\ NonFlier \Leftrightarrow F$$

the proof of which need not mention the predicate, since it suffices to know that the subject does not exist.

This general claim can be expressed in HOL and proven. To make it read sensibly however, it is best for us to define the relevant notion of existence.

HOL Constant

$$\boldsymbol{Existent} : 'a \rightarrow BOOL$$

$$\forall x\bullet\ Existent\ x \Leftrightarrow T$$

The definition looks a bit bizarre and works for just the same reason that we do not need to know the predicate to know that predication to a non-existent will be false. This predicate will always be true, unless it is applied to a term which fails to denote. So we satisfy another of Grice's desiderata, that the non-existence of Pegasus can be stated.

But first a more general result:

$$\boldsymbol{Existent\_lemma\_1} =$$
$$\vdash \forall\ P\ x\bullet \neg\ x\ is_c\ Existent \Rightarrow \neg\ x\ is_c\ P$$

Desideratum A8 is satisfied by the standard quantifiers in HOL.

$$\exists\boldsymbol{\_lemma\_1} =$$
$$\vdash \forall\ P\bullet\ Pegasus\ is_c\ P \Rightarrow (\exists\ x\bullet\ x\ is_c\ P)$$

$$\forall\boldsymbol{\_lemma\_1} =$$
$$\vdash \forall\ P\bullet\ (\forall\ x\bullet\ x\ is_c\ P) \Rightarrow Pegasus\ is_c\ P$$

7.4.1 Quantifier Inference-Rules

An example of the effect of substitutions obtained by use of these theorems in C intended to mimic the rules of Q is as follows.

Take ths simple case of $\forall E$, with the predicate $\lambda x \bullet\ \neg\ x\ is_c\ Flier$, which we seek to instantiate to Pegasus.

We can specialise $\forall E$ to this case like this:

SML

$$val\ sve = list\_\forall\_elim\ [\ulcorner[]{:}BOOL\ LIST\urcorner, \ulcorner\lambda x\bullet\ \neg\ x\ is_c\ Flier\urcorner, \ulcorner Pegasus\urcorner]\ \forall E;$$

This gives the theorem:

$$val\ sve = \vdash ([] \vDash (\forall\ x\bullet\ (\lambda\ x\bullet \neg\ x\ is_c\ Flier)\ x))$$
$$\Rightarrow ([] \vDash (\lambda\ x\bullet \neg\ x\ is_c\ Flier)\ Pegasus) : THM$$

in which there are two occurrences of the lambda expression in the places previously occupied by the variable P.

The substitutions can then be effected by beta reduction which can be achieved by rewriting the theorem as follows:

```
SML
|rewrite_rule [] sve;
```

Which yields a theorem justifying the required inference.

```
|val it = ⊢ ([] ⊨ (∀ x• ¬ x is_c Flier)) ⇒ ([] ⊨ ¬ Pegasus is_c Flier) : THM
```

All the above rules simply explicate the standard aspects of the logic of Q and C and do not reflect the specific additional features which Grice requires in connection with vacuous names. They simply illustrate that the logical context for C includes as much standard logic as is provided in Q, and provide some tenuous indication that Grice's rules conform to standard logic as defined in Church's simple theory of types. The main risk for Grice is in the detail of his presentation, bearing in mind the additional complexities arising from his suffix notation, and the above exercise offers no assurance in that matter.

7.4.2 Existence

A. (E Committal)

We have already defined the predicate "Existent" which tells us whether a name designates.

The following theorem indicates that the propositional function captures the intent of Grice's "+exists" notation.

```
|E1 =⊢ ∀φ α• α is_c φ ⇒ α is_c Existent
```

E Committal Grice's notion of E committal corresponds in System C to the previously defined generic notion of committal instantiated to the propositional function *Existent*.

The definition we have already given for *Existent* is similar in character and identical in effect to Grices suggestions for *+exists*.

B. Existentially Quantified Formulae

This has already been covered.

Grice wishes to distinguish himself from Meinong, but his system allows quantification over non-existents, and he can therefore only distinguish himself from Meinong by distancing himself also from Quine's criterion of ontological committment. This he could easily do by adopting Carnap's distinction between internal and external questions, and denying that any metaphysical significance attaches to something being counted in the range of quantification.

7.4.3 Identity

HOL already has identity and this identity is good for System C. In HOL identity is a curried propositional function of polymorphic type ⌜:'a → 'a → BOOL⌝. If applied using function application the universal law of reflexivity holds.

To get a "strict" version of equality (i.e. one which is false when either argument does not exist) it is necessary to uncurry equality so that its type becomes ⌜:('a × 'a) → BOOL⌝ and apply it using the system C copula for binary relations.

These two seem to correspond to Grice's intended two kinds of equality, insofar as the difference between the two is primarily that one does and the other does not allow that there may be true identities between non-denoting names. However this is achieved in system C by having two quite distinct identity relations only one of which is properly a predicate, and applied through our copula is_c. The other is a propositional function which must be applied without use of the copula (because the copula forces strictness, i.e. forces predication to undefined values to yield falsehoods).

Grice seems to think that the required distinction between equalities can be achieved by the use of his scoping subscripts, but I have not yet been convinced that he is correct in this. Whatever the scope of the predicate, predication to non-denoting names must yield falsehood, so I can't see how any use of subscripts will make "Pegasus = Pegasus" to be true.

To illustrate the effect of the built-in equality the following theorems suffice:

Pegasus_eq_lemma1 =
⊢ *Pegasus* = *Pegasus*

Note the lack of copula. This is of course an instance of the theorem:

eq_refl_thm =
⊢ ∀ *x*• *x* = *x*

To get a nice notation for "strong" equality we need to define it as an infix symbol, which requires that the copula be built into the definition. We could use the ≡ symbol, but this usually represents an equivalence relation which is weaker than identity, so we will subscript the = sign with c, suggesting that it is just the same thing applied through our copula.

SML
declare_infix (*200*, "$=_c$");

HOL Constant
$\$=_{\mathbf{c}} : 'a \to 'a \to BOOL$

$\forall x\ y{:}'a\bullet\ (x =_c y) = (x,y)\ is_{c2}\ (Uncurry\ \$=)$

We can now prove:

$$\begin{array}{|l} Pegasus\_eq\_lemma2 = \\ \quad \vdash \neg\ Pegasus =_c Pegasus \end{array}$$

And must rest content with the qualified rule:

$$\begin{array}{|l} \boldsymbol{eq_c\_refl\_thm} = \\ \quad \vdash \forall\ x\bullet\ x\ is_c\ Existent \Leftrightarrow x =_c x \end{array}$$

The "strictness" of this equality is expressed by the theorem (which would have made a better definition):

$$\begin{array}{|l} eq_c\_strict\_thm = \\ \quad \vdash \forall\ x\ y\bullet\ x =_c y \Leftrightarrow x\ is_c\ Existent \wedge y\ is_c\ Existent \wedge x = y \end{array}$$

On reflection Grice's treatment by scoping is presented in the context of a second order definition of equality, and therefore for the weak equality does not define equality as a predicate, but as a second order formula. Its equivalence to the primitive equality in HOL can be illustrated by the following theorem:

$$\begin{array}{|l} \boldsymbol{snd\_order\_eq\_thm} = \\ \quad \vdash \forall\ x\ y\bullet\ x = y \Leftrightarrow (\forall\ P\bullet\ x\ is_c\ P \Leftrightarrow y\ is_c\ P) \end{array}$$

This equivalence only holds because we adopted Grice's preferred (but not stipulated) domain of interpretation, in which there is exactly one non-designating correlate. If there was more than one non-designating correlate then Grice's second order definition would give a notion of identity under which all non-designating correlates are considered equal, which would not be the case for the primitive equality in HOL.

Even in that case a similar device would work as illuatrated by the following theorem:

$$\begin{array}{|l} \boldsymbol{snd\_order\_eq\_thm2} = \\ \quad \vdash \forall\ x\ y\bullet\ x = y \Leftrightarrow (\forall\ P\bullet\ P\ x \Leftrightarrow P\ y) \end{array}$$

In which the griceian copula has been dropped so that the predication is no longer strict.

7.4.4 Names and Descriptions

I note here pro-tem just that the method adopted in System C for controlling the scope of predications, viz. the use either of lambda expressions or of naming complex predicates by definitions, enables scope to be controlled in predication to descriptions which are formed as terms. We can therefore follow Grice's system Q in introducing such descriptions.

SML

```
declare_binder "ι";
```

HOL Constant

$$\$\boldsymbol{\iota} : ('a \to BOOL) \to 'a$$

$$\forall\phi\bullet\ (\$\iota\ \phi) = \epsilon\alpha\bullet\ (\alpha\ is_c\ \phi \wedge \forall\beta\bullet\ \beta\ is_c\ \phi \Rightarrow \beta = \alpha)$$
$$\vee\ (\neg\exists\gamma\bullet\ \gamma\ is_c\ \phi \wedge \forall\beta\bullet\ \beta\ is_c\ \phi \Rightarrow \beta = \gamma) \wedge \alpha = \bot$$

$$\boldsymbol{\iota\_thm1} = \vdash \forall\ \gamma\bullet\ \gamma\ is_c\ \phi \wedge (\forall\ \beta\bullet\ \beta\ is_c\ \phi \Rightarrow \beta = \gamma) \Rightarrow \$\iota\ \phi = \gamma$$
$$\boldsymbol{\iota\_thm2} = \vdash \neg\ (\exists\ \gamma\bullet\ \gamma\ is_c\ \phi \wedge (\forall\ \beta\bullet\ \beta\ is_c\ \phi \Rightarrow \beta = \gamma)) \Rightarrow \$\iota\ \phi = \bot$$

There is a difficulty in adopting Grice's reticence to allow existential generalisation on the basis of such a predication. This cannot be supported in system C without changing the underlying semantics.

Grice has effectively split the non-denoting expressions into two, those which correlate with something in the domain of quantification and those which do not. Since our embedding requires all expressions to have a value, we would have to add another kind of entity into the domain of interpretation and define quantifiers which operate over expressions which have a correlate but not over the values of non-denoting descriptions.

This would effect most of the embedding, the whole would have to be reworked (in a routine way). We investigate this possibility in the next section.

7.5 System S

System S contains two minor adjustments to System C which are concerned with the treatment of non-designating names and failing descriptions. In relation to non-designating names, the semantics in System S now allows that they have distinct correlates. In addition a new subdomain is introduced which provides at least one (since we can always prove that there is a failing description) surrogate for a failing description. The primary reason for recognising, in the semantics, denotations for failing descriptions, is so that we can arrange as Grice seems to require, that they do not fall into the range of the System S quantifiers. This in turn results in the qualification of the rules EG and UI.

A new theory is needed which I will call "griceS" which is created here:

SML

```
open_theory "grice";
force_new_theory "griceS";
set_pc "rbjmisc";
```

We liberalise the notion of interpretation from system C in the following two ways. Firstly we allow that there may me more than one non-denoting correllate. and hence then there may be semantic distinctions between non-denoting names, overcoming one obstacle to substantial discussions about fictional entities. Secondly we introduce values to stand for non-referring descriptions. These are not intended to be values and will be excluded from the range of quantification, with the effect that EG and UI will not work with failing descriptions.

This will be done using a "disjoint union" the right disjunct of which will be surrogates for failing descriptions, and the left will be correlates for names of which there will be at least one which is not a denotatum. The "normal" quantifiers for system S will quantify only over the correlates.

To effect this we introduce the property of being a denotatum which is left open except to the extent that there must be at least one correlate which is not a denotatum.

HOL Constant

$$\boldsymbol{denotatum}: 'a \to BOOL$$

$$\exists x \bullet \neg\ denotatum\ x$$

HOL Constant

$$\bot\_\boldsymbol{corr}: 'a$$

$$\neg\ denotatum\ \bot\_corr$$

The type of exressions in System S is given by the following type abbreviation:

SML

$$declare\_type\_abbrev(\texttt{"}SC\texttt{"},\ [],\ \ulcorner :'a\ +\ 'b \urcorner);$$

HOL Constant

$$\bot: SC$$

$$\bot = InL\ \bot\_corr$$

It will be useful to have tests and projections for values of type ⌜:SC⌝.

This propositional function tests whether an SC is a correlate:

HOL Constant

$$\boldsymbol{IsCorr}: SC \to BOOL$$

$$\forall v{:}SC \bullet\ IsCorr\ v \Leftrightarrow IsL\ v$$

This function extracts the correlate (of type ⌜:'a⌝) from a value of type ⌜:SC⌝.

HOL Constant

$$\boldsymbol{Corr}\text{: } SC \to 'a$$

$$\forall v{:}SC\bullet\ Corr\ v = OutL\ v$$

This propositional function tells us whether a value of type ⌜:SC⌝ is a denotatum.

HOL Constant

$$\boldsymbol{IsDen}\text{: } SC \to BOOL$$

$$\forall v{:}SC\bullet\ IsDen\ v \Leftrightarrow IsCorr\ v \wedge denotatum\ (Corr\ v)$$

There are now difficulties in the treatment of constants and variables, since a HOL constant or variable constrained only by the type of the elements of an interpretation will not be known to denote (which is OK), but will not even be known to correlate, and hence may not even be in the range of quantification. Such results as can be obtained using unconstrained variables will nevertheless be valid, and will be generalisable.

Now we define our special kind of predication (the copula for System S).

SML

```
declare_infix (400, "is_s");
```

HOL Constant

$$\$is_s\text{: } SC \to ('a \to BOOL) \to BOOL$$

$$\forall\ p\ t\bullet\ t\ is_s\ p \Leftrightarrow if\ IsDen\ t\ then\ p\ (Corr\ t)\ else\ F$$

The definition says, if the term denotes then apply the predicate to the value denoted, otherwise the result of the predication is F (false).

Note that this will not work for relations, we would have to define a separate similar predicator for each arity of relation in use.

Here is the copula for applying a 2-ary relation.

SML

```
declare_infix (400, "is_s2");
```

HOL Constant

$\$is_{s2}$: $(SC \times SC) \to ('a \times 'a \to BOOL) \to BOOL$

$\forall\ p\ t\bullet\ t\ is_{s2}\ p \Leftrightarrow$ *if* $IsDen\ (Fst\ t) \wedge IsDen\ (Snd\ t)$
then $p\ (Corr\ (Fst\ t),\ Corr\ (Snd\ t))$
else F

Now let's introduce Pegasus. The "definition" just says that Pegasus is a non-denoting correlate. We know that there must be such a thing (though we don't know that there are any denoting correlates).

HOL Constant

$\boldsymbol{Pegasus} : SC$

$IsCorr\ Pegasus \wedge \neg\ IsDen\ Pegasus$

And the predicate *Flier*. Note that the predicates apply only to real desigata under the normal HOL application, its only under our special predication that they become applicatble to possibly non denoting values.

HOL Constant

$\boldsymbol{Flier} : 'a \to BOOL$

T

We can now prove in HOL the elementary facts we know about Pegasus flying, yielding the following theorems:

Pegasus_lemma_1 =
$\vdash Pegasus\ is_s\ Flier \Leftrightarrow F$

Pegasus_lemma_2 =
$\vdash Pegasus\ is_s\ (\lambda\ x\bullet \neg\ Flier\ x) \Leftrightarrow F$

Pegasus_lemma_3 =
$\vdash \neg\ Pegasus\ is_s\ Flier$

Pegasus_lemma_4 =
$\vdash \neg\ Pegasus\ is_s\ (\lambda\ x\bullet \neg\ Flier\ x)$

In case you may doubt the neutrality of our use of lambda expressions, we show here that they are inessential to this approach, and can be dispensed with in favour of explict definitions of the predicates required. Thus we may instead define a predicate which is the denial of *Flier*:

HOL Constant

$$\boldsymbol{NonFlier} : 'a \rightarrow BOOL$$

$$\forall x\bullet\ NonFlier\ x \Leftrightarrow \neg\ Flier\ x$$

We can then prove the slightly reworded:

$$\boldsymbol{Pegasus\_lemma\_5} = \vdash\ Pegasus\ is_s\ NonFlier \Leftrightarrow F$$

the proof of which need not mention the predicate, since it suffices to know that the subject does not exist.

This general claim can be expressed in HOL and proven. To make it read sensibly however, it is best for us to define the relevant notion of existence.

HOL Constant

$$\boldsymbol{Existent} : 'a \rightarrow BOOL$$

$$\forall x\bullet\ Existent\ x \Leftrightarrow T$$

The definition looks a bit bizarre and works for just the same reason that we do not need to know the predicate to know that predication to a non-existent will be false. This predicate will always be true, unless it is applied to a term which fails to denote. So we satisfy another of Grice's desiderata, that the non-existence of Pegasus can be stated.

But first a more general result:

$$\boldsymbol{Existent\_lemma\_1} =$$
$$\vdash\ \forall\ P\ x\bullet\ \neg\ x\ is_s\ Existent \Rightarrow \neg\ x\ is_s\ P$$

Desideratum A8 is satisfied by the standard quantifiers in HOL.

$$\exists\boldsymbol{\_lemma\_1} =$$
$$\vdash\ \forall\ P\bullet\ Pegasus\ is_s\ P \Rightarrow (\exists\ x\bullet\ x\ is_s\ P)$$

$$\forall\boldsymbol{\_lemma\_1} =$$
$$\vdash\ \forall\ P\bullet\ (\forall\ x\bullet\ x\ is_s\ P) \Rightarrow Pegasus\ is_s\ P$$

However, the HOL native quantifiers are unsatisfactory since they quantify over the values introduced into the domain of the interpretation as correlates for failing descriptions.

This is shown by the following theorem:

$$\exists\boldsymbol{\_not\_Corr\_thm} =$$
$$\vdash\ \exists\ x\bullet\ \neg\ IsCorr\ x$$

7.5.1 Quantifier Definitions

We now define quantifiers which quantify over correlates only, not over the values of failing descriptions.

Quantifiers are propositional functions over propositional functions, and are usually syntactically "binders", so that they are expected to be applied to a lambda abstraction and the lambda in that expression is to be omitted. We use the usual quantifier symbols subscripted with s to distinguish them from the standard quantifiers.

SML

$declare\_binder$ "$\forall_s$";
$declare\_binder$ "$\exists_s$";

The domain of interpretation now includes correlates and some other things (for failing descriptions). The things for failing descriptions are to be excluded from the range of the quantifiers (they really really don't exist) so quantifiers range only over correlates. The dollar sign locally supresses the binder status for the purpose of giving the definition.

HOL Constant

$$\$\forall_{\boldsymbol{s}} : (SC \to BOOL) \to BOOL$$

$$\forall f \bullet \ \$\forall_s \ f \Leftrightarrow \forall x \bullet \ IsCorr \ x \Rightarrow f \ x$$

HOL Constant

$$\$\exists_{\boldsymbol{s}} : (SC \to BOOL) \to BOOL$$

$$\forall f \bullet \ \$\exists_s \ f \Leftrightarrow \exists x \bullet \ IsCorr \ x \wedge f \ x$$

To express the rules for these new quantifiers we define a new notion of existence. This cannot be a bona-fide predicate, since such a predicate would of necessity be false of the correlates which are not designata (if applied through the copula), so it must be thought of as a propositional function to be applied without the copula.

HOL Constant

$$\boldsymbol{Existent_s} : SC \to BOOL$$

$$\forall x \bullet \ Existent_s \ x \Leftrightarrow \exists_s \ y \bullet \ x = y$$

7.5.2 Quantifier Inference-Rules

The propositional rules are common to Systems C and S and are shown above in Section 7.3.4.

We have new quantifiers in System S, and the quantifier rules will be modified accordingly.

$$\begin{array}{|ll}
\forall \boldsymbol{I_s} = & \vdash \forall\ \Gamma\ P\bullet\ (\forall\ x\bullet\ \Gamma \vDash \mathit{Existent}_s\ x \wedge P\ x) \Rightarrow (\Gamma \vDash (\forall_s\ x\bullet\ P\ x)) \\
\forall \boldsymbol{E_s} = & \vdash \forall\ \Gamma\ P\ c\bullet\ (\Gamma \vDash \mathit{Existent}_s\ c \wedge (\forall_s\ x\bullet\ P\ x)) \Rightarrow (\Gamma \vDash P\ c) \\
\exists \boldsymbol{I_s} = & \vdash \forall\ \Gamma\ P\ x\bullet\ (\Gamma \vDash \mathit{Existent}_s\ x \wedge P\ x) \Rightarrow (\Gamma \vDash (\exists_s\ x\bullet\ P\ x)) \\
\exists \boldsymbol{E_s} = & \vdash \forall\ P\ Q\ c\bullet\ (\forall\ x\bullet\ [\mathit{Existent}_s\ x \wedge P\ x] \vDash Q) \Rightarrow ([\exists_s\ x\bullet\ P\ x] \vDash Q)
\end{array}$$

An example of the effect of substitutions obtained by use of these theorems in S intended to mimic the rules of Q is as follows.

Take ths simple case of $\forall E_s$, with the predicate $\lambda x \bullet\ \neg\ x\ is_s\ Flier$which we seek to instantiate to Pegasus.

We can specialise $\forall E_s$ to this case like this:

```
SML
val sves = list_∀_elim [⌜[]:BOOL LIST⌝, ⌜λx: 'a + 'b• ¬ x is_s Flier⌝, ⌜Pegasus⌝] ∀E_s;
```

This gives the theorem:

$$\begin{array}{|l}
\mathit{val}\ \mathit{sves} = \vdash ([] \vDash \mathit{Existent}_s\ \mathit{Pegasus} \wedge (\forall_s\ x\bullet\ (\lambda\ x\bullet \neg\ x\ \mathit{is}_s\ \mathit{Flier})\ x)) \\
\qquad\qquad \Rightarrow ([] \vDash (\lambda\ x\bullet \neg\ x\ \mathit{is}_s\ \mathit{Flier})\ \mathit{Pegasus}) : \mathit{THM}
\end{array}$$

in which there are two occurrences of the lambda expression in the places previously occupied by the variable P.

The substitutions can then be effected by beta reduction which can be achieved by rewriting the theorem as follows:

```
SML
rewrite_rule [] sves;
```

Which yields a theorem justifying the required inference.

$$\begin{array}{|l}
\mathit{val}\ \mathit{it} = \vdash ([] \vDash \mathit{Existent}_s\ \mathit{Pegasus} \wedge (\forall_s\ x\bullet\ \neg\ x\ \mathit{is}_s\ \mathit{Flier})) \\
\qquad\qquad \Rightarrow ([] \vDash \neg\ \mathit{Pegasus}\ \mathit{is}_s\ \mathit{Flier}) : \mathit{THM}
\end{array}$$

7.5.3 Existence

A. (E Committal)

We have already defined the propositional function “Existent” which tells us whether a name designates.

The following theorem indicates that the propositional function captures the intent of Grice’s “+exists” notation.

$\vdash \forall\ \phi\ \alpha\bullet\ \alpha\ is_s\ \phi \Rightarrow Existent\ \alpha$ **E1**

Once again, Grice's notion of *E_committal* is obtainable for System S by instantiation of our generic notion of *committal* by the propositional function *Existent*.

The definition we have already given for *Existent* is similar in character and identical in effect to Grices suggestions for *+exists*.

B. Existentially Quantified Formulae

This has already been covered.

Grice wishes to distinguish himself from Meinong, but his system allows quantification over non-existents, and he can therefore only distinguish himself from Meinong by distancing himself also from Quine's criteria of ontological committment. This he could easily do by adopting Carnap's distinction between internal and external questions, and denying that any metaphysical significance attaches to something being counted in the range of quantification.

7.5.4 Identity

HOL already has identity and this identity is good for System C. It HOL it is a curried propositional function of polymorphic type ⌜:'a → 'a → BOOL⌝. If applied using function application the universal law of reflexivity holds.

To get a "strict" version of equality (i.e. one which is false when either argument does not exist) it is necessary to uncurry equality so that its type becomes ⌜:('a × 'a) → BOOL⌝ and apply it using the system C copula for binary relations.

These two seem to correspond to Grice's intended two kinds of equality insofar as the difference between the two is primarily that one does and the other does not allow that there may be true identities between non-denoting names. However this is achieved in system C by having two quite distinct identity relations only one of which is properly a predicate, and applied through our copula is_s. The other is a propositional function which must be applied without use of the copula (because the copula forces strictness, i.e. forces predication to undefined values to yield falsehoods).

Grice seems to think that the required distinction between equalities can be achieved by the use of his scoping subscripts, but I have not yet been convinced that his is correct in this. Whatever the scope of the predicate, predication to non-denoting names must yield falsehood, so I can't see how any use of subscripts will make "Pegasus = Pegasus" true.

To illustrate the effect of the built in equality the following theorems suffice:

***Pegasus_eq_lemma*1** =
$\vdash Pegasus = Pegasus$

Note the lack of copula. This is of course an instance of the theorem:

$$\begin{array}{|l} \boldsymbol{eq\_refl\_thm} = \\ \quad \vdash \forall\ x\bullet\ x = x \end{array}$$

To get a nice notation for "strong" equality we need to define it as an infix symbol, which requires that the copula be built into the definition. We could use the $\equiv$ symbol, but this usually represents an equivalence relation which is weaker than identity, so we will subscript the = sign with g, suggesting that it is just the same thing applied through our copula.

SML

$$\begin{array}{|l} declare\_infix\ (200,\ "=_s"); \end{array}$$

HOL Constant

$$\begin{array}{|l} \$\boldsymbol{=_s} : SC \to SC \to BOOL \\ \hline \\ \forall x\ y{:}SC\bullet\ (x =_s y) = (x,y)\ is_{s2}\ (Uncurry\ \$=) \end{array}$$

We can now prove:

$$\begin{array}{|l} Pegasus\_eq\_lemma2 = \\ \quad \vdash \neg\ Pegasus =_s Pegasus \end{array}$$

And must rest content with the qualified rule:

$$\begin{array}{|l} \boldsymbol{eq_s\_refl\_thm} = \\ \quad \vdash \forall\ x\bullet\ x\ is_s\ Existent \Leftrightarrow x =_s x \end{array}$$

The "strictness" of this equality is expressed by the theorem (which would have made a better definition):

$$\begin{array}{|l} eq_s\_strict\_thm = \\ \quad \vdash \forall\ x\ y\bullet\ x =_s y \Leftrightarrow x\ is_s\ Existent \wedge y\ is_s\ Existent \wedge x = y \end{array}$$

The primitive equality in HOL no longer corresponds to Grice's second order definition and the proof of *snd_order_eq_thm* fails.

$$\begin{array}{|l} \boldsymbol{neg\_snd\_order\_eq\_thm} = \\ \quad \vdash \neg\ (\forall\ x\ y\bullet\ x = y \Leftrightarrow (\forall\ P\bullet\ x\ is_s\ P \Leftrightarrow y\ is_s\ P)) \end{array}$$

There are two reasons for this. The first is that there is now more than one possible non-denoting correlate, and Leibniz's formula, applied through the System S copula does not distinguish between them. The second is the presence of values for failing descriptions.

The values for failing descriptions should not be in the range of quantification, so we do need to have special quantifiers in System S, it will then also be necessary to qualify the quantification to get a version of the second order definition which takes account of possibly distinct non-designating names.

This equivalence fails because we now allow (in fact, require) more than one non-designating value in the domain of an interpretation.

However, if the copula is dropped we still get a working definition of a 'weak' equality.

$$\begin{array}{l} \boldsymbol{snd\_order\_eq\_thm2} = \\ \quad \vdash \forall\ x\ y\bullet\ x = y \Leftrightarrow (\forall\ P\bullet\ P\ x \Leftrightarrow P\ y) \end{array}$$

7.6 Conclusions

Chapter 8

Conclusions

Index

$=_c$ . . . 144
$=_s$ . . . 154
$\Diamond$ . . . 66, 78, 102
$\Diamond AarenotA\_thm$ . . . 108
$\Diamond AllBareA\_thm$ . . . 107
$\Diamond\_conv$ . . . 83, 107
$\Diamond\_intro\_thm$ . . . 84, 108
$\Diamond_a$ . . . 66
$\Leftrightarrow_a$ . . . 80, 104
$\Rightarrow_a$ . . . 80, 104
$\Vdash$ . . . 81, 106
$\bot$ . . . 139, 147
$\bot\_corr$ . . . 147
$\exists E$ . . . 138
$\exists E_s$ . . . 152
$\exists I$ . . . 138
$\exists I_s$ . . . 152
$\exists\_lemma\_1$ . . . 142, 150
$\exists\_not\_Corr\_thm$ . . . 150
$\exists_a$ . . . 81, 105
$\exists_s$ . . . 151
$\forall E$ . . . 138
$\forall E_s$ . . . 152
$\forall I$ . . . 138
$\forall I_s$ . . . 152
$\forall\_lemma\_1$ . . . 142, 150
$\forall_a$ . . . 80, 104
$\forall_s$ . . . 151
ι . . . 146
ι_thm1 . . . 146
ι_thm2 . . . 146
$\wedge E$ . . . 137
$\wedge I$ . . . 137
$\wedge_a$ . . . 79, 103
$\neg_a$ . . . 79, 103
$\neg_m$ . . . 67
$\vee E$ . . . 137
$\vee I$ . . . 137
$\neq$ . . . 12
$\vDash$ . . . 66, 81, 105

a . . . 46, 51, 55, 60, 65, 96
$ACAT$ . . . 16
$accidentally\_predicable\_of$ . . . 20
$actual\_world$ . . . 66, 74, 101
ai_conv_thm . . . 57, 62, 98
All . . . 77, 102
$All\_are\_izz\_or\_hazz\_lemma$ . . . 78
$All\_are\_not\_lemma$ . . . 78
are . . . 78, 102
are_conv1 . . . 107
are_conv2 . . . 107
are_conv3 . . . 107
$are\_izz\_neq\_hazz\_lemma$ . . . 78
are_not_lemma . . . 106
$Aristotle$ . . . **12**
  $organon$ . . . 12
ASS . . . 137
$AttrSet$ . . . 17

CAT . . . 16
Cat . . . 17
$category$ . . . 75
$Category\_of\_Substance$ . . . 74
$CATM$ . . . 16
$CatSet$ . . . 17
$CatSubs$ . . . 16
$Code$ . . . 12
$committal$ . . . 138
$Complement$ . . . 48, 57, 62, 98
$Corr$ . . . 148
CP . . . 137

$denotatum$ . . . 147
$difficulties$ . . . 131
DN . . . 137

e . . . 46, 51, 56, 60, 65, 96
$E1$ . . . 143, 153
e_conv . . . 52
e_conv_thm . . . 57, 62, 98
eo_conv_thm . . . 57, 62, 98
eq_refl_thm . . . 144, 154
$eq_c\_refl\_thm$ . . . 145
$eq_s\_refl\_thm$ . . . 154
$essentially\_predicable\_of$ . . . 20
$Existent$ . . . 142, 150
$Existent\_lemma\_1$ . . . 142, 150
$Existent_s$ . . . 151
$extension$ . . . 100

$figureS$ . . . 32
$figurest$ . . . 33

Flier 141, 149
form 22
*FP*1 86
*FP*2 86
*FP*3 86
*FP*3*b* 86
*FP*4 86
*FP*4*a* 86
*FP*4*c* 87

hazz 18, 78
*hazz_conv*2 83

I 64
i 46, 51, 56, 60, 65, 96
i_conv 52
i_conv_thm 57, 62, 98
inclinations 131
Individual 87
individual 21, 86, 108
IndvSet 17
InTermM 77
IsCorr 147
IsDen 148
IsPos 96
ISUB 16
izz 18, 77
*izz_conv*1 82
*izz_conv*2 82
*izz_conv*3 82
izz_not_lemma 82

leibniz_identity 94

map_goal 69
mk_AttrTerm 75
mk_modsyll 69
mk_modsyllp 69
mk_modt 69
mk_pred 68
mk_relt 68
mk_SubsTerm 75
mk_syll 68
mk_syllp 68
mkATvar 33
mkGoals 35
mkSimpMapkit 36
mkSyll 35
modgoal 69
mods_mapkit 54
Mp 86
MPP 137

neg_snd_order_eq_thm 154
NonFlier 141, 150
not 76, 102
not_hazz_not_lemma 83
not_some_hazz_lemma 83
nthfig 33

objectives 132
  detailed 132
op_from_char 36
optrip_from_text 36

p 51, 65
*P_*2 139
*P_*3 139
*P_*4 139
*P_*5 139
*P_*6∃*b* 139
*P_*6∀ 139
particular 21, 88
Pegasus 140, 149
*Pegasus_eq_lemma*1 ..... 144, 153
*Pegasus_lemma_*1 ..... 141, 149
*Pegasus_lemma_*2 ..... 141, 149
*Pegasus_lemma_*3 ..... 141, 149
*Pegasus_lemma_*4 ..... 141, 149
*Pegasus_lemma_*5 ..... 141, 150
Plato 12
predicable_of 21
proveGoals 35
proveSylls 35

RAA 137
resolutions 132

*sg_*03 53
*sg_*04 53
*sg_*09 53
*sg_*10 53
*sg_*11 53
*sg_*12 53
*sg_*13 53
*sg_*14 53
snd_order_eq_thm 145
*snd_order_eq_thm*2 ..... 145, 155
Some 77, 96, 102
Some_are_izz_or_hazz_lemma 78

some_are_lemma 106
some_izz_lemma 82
SomeThing 22
STT 134
Substance 75
substantial 88
syll_prove 69
syll_prove_and_store 69
syll_rule 47
*syll_rule*2 52
*syll_rule*3 57
*syll_rule*6 61
*syll_rule*6*b* 61
syll_ruleL 98
syll_ruleLb 98
syll_tac 47
*syll_tac*2 52
*syll_tac*3 57
*syll_tac*6 61
*syll_tac*6*b* 61
syll_tacL 98
syll_tacLb 98
*syllogism_data*1 34
*syllogism_data*2 34
*syllogism_data*3 34

tag 75
TermM 75

u 47, 52, 56, 60, 66, 97
u − *p* 28, 34
universal 21, 88
universal − *particular* 28, 34

vowels_from_string 35

W 64

Id: b001.tex,v 1.13 2011/01/05 11:02:31 rbj Exp
Id: t028.doc,v 1.33 2012/01/23 21:40:02 rbj Exp
Id: t032.doc,v 1.4 2012/02/14 20:44:22 rbj Exp
Id: t030.doc,v 1.9 2015/04/23 09:58:07 rbj Exp