Analyses of Analysis: Part II - Synthetic Analysis

Roger Bishop Jones

Created 2009/07/22

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

Id: b003.tex,v 1.10 2011/01/05 11:02:31 rbj Exp

Contents

Chapter 1

Introduction

This is a kind of *analytic* history of philosophical analysis. It is intended to be analytic in a specific sense, in adopting a method of *logical analysis* which is articulated as a central part of the philosophical posture which I have called *Metaphysical Positivism*[**?**]. We may think of the history as proceeding by the construction of *rational reconstructions* of various historical conceptions of philosophical analysis and related matters. The adoption of the cited method of logical analysis determines the way in which these rational constructions are undertaken.

This results in the rational constructions being undertaken by the construction of abstract models. Though it is not essential to the method, these abstract models are defined using formal notations. The formal specifications of these models, and various proofs in relation to them, are checked by computer in various respects (syntactic well formedness and type correctness of terms and formulae, correctness of proofs).

In presenting these, I have attempted to provide for readers approaching at three different levels.

At the first level, it is hoped that the main philosophical ideas will be intelligible independently of the formal material which provides the detail in the various analyses. At the second level, a reading of the formal materials, consisting of definitions and theorems (but not the proofs), will provide detailed support for the philosophical conclusions and will permit a greater depth of understanding of the issues. A third level is available for anyone wishing to understanding the details of proofs, to further progress the analysis in any area, or to address other problems using the same methods and tools.

For a reading at the first level it is intended that it will suffice to read the first and last chapters of each part and the first and last sections of other chapters, omitting part three in its entirety. For a reading at the second level the entire text is relevant, though many parts are independent and a detailed account of the dependencies will be supplied. At the third level the reader should install the software used for this development and the scripts which constitute the formal part of the book, make reference to various other documents, and acquire a full understanding of the methods and results by interaction with the proof support software.

Readers hoping eventually for an in depth understanding might possibly find it useful to read through at level one first, and then take in the formal material on a second pass. Alternatively this plan might be adopted chapter by chapter, or exclusively for those particular topics which may interest them.

In this introductory chapter Sections 1.1 to 1.3 are intended for readers at all levels. To understand the formal materials it will also be necessary to read Section 1.4. For hands on work see Section 1.5.

1.1 Philosophical Introduction

This analytic history of philosophical analysis is written with a specific terminus in mind, the method of lgical analysis which comes with *Metaphysical Positivism*[**?**]. The perspective from which the history is viewed is that of Metaphysical Positivism, the concepts and methods used in the analysis come from the same source, the history as a whole is offered as an extended exemplar of the methods, and involves an exposition of the methods, their philosophical underpinnings and their historical origins.

In this introductory material I hope to provide a sufficient introduction to the methods so that the reader will be able to understand what follows, and to give a description of the coverage of the material which follows and how it fits into the story which I seek to present.

The work is not a presentation of wisdom already realised, and is not a scholarly exercise in the normal sense which is now associated with that phrase. It is a record of an exploration intended to cast light on the origins of a philosophical viewpoint which I have acquired during the course of a lifetime in which among other preoccupations these matters have featured. The application of the proposed method to historical investigations is new, and it remains to be seen how much light it casts.

1.1.1 Some Preliminary Philosophical Orientation

The epicentre of *Metaphysical Positivism* is the distinction between *truths of reason* and *matters of fact* which is sometimes known as *Hume's Fork*. Truths of reason are also called *logical truths*, *analytic truths*, and *logically necessary truths*, all these being synonymous in the conceptual scheme of *metaphysical positivism*.

A discussion of the history of philosophical logic will involve us in consideration of diverse usage of these and other important terms. We therefore need a method which allows us to separate our language from that of the various historical figures who feature in the history.

This method is "The Method of Formal Logical Analysis", which makes use of formal abstract models of the theories of philosophers to provide precise analyses of their usage.

1.2 Formal Methods

1.2.1 Formal Methods in Philosophy

At least three kinds of work are described by philosophers as constituting "formal philosophy" [1] .

1. the use of mathematics in addressing philosophical problems
2. the use of metamathematics or meta-logic in addressing philosophical problems
3. the conduct of philosophy using formal deductive arguments where possible.

Of these the first two dominate.

I will expand this discussion, but this document will primarily be concerned with explaining one approach to item 3.

[1] This discussion is primarily based on the accounts by philosophers of their involvement in formal philosophy published in [?, ?].

1.2.2 Formal Methods in Information Engineering

1.3 An Overview of Part I

This is a rather selective history focusing on developments which are significant in the history of ideas which underpins the formal analytic method under consideration, and which prove interesting in the light of that kind of analysis.

The most central topic in the first volume is that of logical truth, which itself is not often directly addressed, but which is related to quite a number of other concepts.

Some precursors of our present conception of logical truth are:

Plato	The distinction between the world of forms and the world of appearances
Aristotle	The syllogism and modal syllogism and the distinction between necessary and contingent truth, the distinction between essential and accidental and the metaphysics on which that depends.
Leibniz	Leibniz's conception of the distinction between necessary and contingent truth, his conception of analysis, the ideas behind his *calculus ratiocinator*.
Hume	Hume's fork.
Kant	Kant's conceptions of analyticity and necessity.
Frege	Semantics and Logic.
Wittgenstein	The conception of logical truth as tautological and the metaphysics of logical atomism.

The provisional list of chapters is:

1. Introduction (this document [**?**])
2. Plato and Aristotle [**?**]
3. Leibniz [**?**]
4. Hume and Kant [**?**]
5. Frege [**?**]
6. Russell and Wittgenstein [**?**]

I would like to maximise the connectedness of this, and in this I see a couple of spines of development. The first is Aristotle-Leibniz-Russell. The second perhaps Plato-Hume-Frege-Wittgenstein. I don't know whether much can be made of this, the inter-relationship are complex, but some simplified view may contribute to their clarification.

1.4 Understanding The Formal Models

1.4.1 Languages

Two formal languages are used in this document. One is a language, called HOL for Higher Order Logic, with a deductive system sufficient for the formal development of mathematics, or for other applications susceptible of formalisations.

The other is a metalanguage known as SML, which stands for Standard Meta-Language. This is used for giving executive instructions to software which provides support for use of the HOL language and for the construction and verification of formal proofs in the HOL logic.

To understand the specifications it is not necessary to understand the metalanguage SML, it is used in the specifications only for very elementary purposes (typically for introducing a new type or controlling some aspect of the concrete syntax) which we can be understood without any general understanding of the metalanguage. If it comes to doing proofs, a fuller understanding of the metalanguage is desirable and more information is provided in Section 1.5.

We therefore confine ourselves here to explaining the specification language HOL, a variety of Higher Order Logic. To understand specifications in HOL the reader needs to know about the *semantics* of HOL and that is what I aim to present here.

1.4.2 Abstract Ontology

The first thing we need to understand in relation to the semantics is what the language is about, and this may be read as concerning the underlying ontology. There is no single answer to this question, but the method which we have adopted and of whose description this account of HOL forms a part, is a a method based on abstract modelling. We therefore present HOL as a language for talking about pure well-founded sets.

1.4.3 HOL Types

HOL is a type theory, based a simply typed lambda calculus.

As such you may think in the first instance of the types as consisting of a type of individuals, a type of propositions, and a hierarchy of types obtained by forming functions spaces from available types. In Russell's theory of types the individuals were intended as concrete and contingent entities so that the necessary axiom of infinite was held even by Russell to be contingent. In our case we consider the individuals to be some infinite collection of abstract entities, and the axiom of infinity is not contingent.

1.4.4 HOL Terms

The principle language used here (apart from English) is a language (and logic) called HOL. HOL is an acronym for Higher Order Logic, of which there are many different varieties, and is also widely used for a specific variant of higher order logic which has been implemented in several computer programs providing support for formal specification and proof.

For full details of this language you would need to refer to the documentation which comes with these tools [**?**] or some of the papers published about them. See below (Section 1.5.2) for information relating to the tool used for producing this document, ProofPower-HOL.

The language HOL is a direct descendent of Russell's *Theory of Types*[**?**, **?**], the logic which he and Whitehead used in *Principia Mathematica*[**?**]. To get from Russell's *Theory of Types* to HOL you do the following (names in brackets give credit to the person who thought of the step):

- discard the ramifications (Ramsey [**?**])
- simplify by basing on typed lambda-calculus (Church [**?**])
- add polymorphism (Gordon/Milner [**?**, **?**])

To do serious work you need a proof tool, see (Section 1.5.2).

The following are the most important features of this language/logic which distinguish it from those typically considered by philosophers.

- It is a foundation system, i.e. it suffices for the development of mathematics by conservative extension (definitions) alone.
- It has a type system, and allows new types to be defined.
- It is supported by computer software which checks specifications, assists in constructing proofs and rigorously checks proofs.

1.4.5 Methods

The principle technique used here is a method which has some of the theoretical merits of a meta-theoretic treatment, but is less arduous and provides better support for reasoning in the object language.

We imagine ourself devising a formal language in which to talk about Aristotle's metaphysics, and in which to formalise the kind of metaphysical arguments which are found in Aristotle. To do this rigorously, we need to deal first with the semantics of the language, and establish a deductive system which is sound with respect to that semantics.

A standard formal treatment of this material would involve a specification of the syntax of an appropriate language, the development of semantics, probably as some kind of model theory, the specification

of a deductive system for the language and a proof of soundness of that system (this would be a version of Aristotle's Syllogistic logic). This is feasible with the languages and tools we are using, but arduous. The results would be good for metatheory, but not necessarily convenient for conducting proofs in the language thus defined, i.e. for reasoning in the new object language.

There is another manner of proceeding which better suits our present purposes. This consists in extending our already available language, using the definitional facilities and the flexibility in its syntax (e.g. fixity declarations) to create a language extension which looks something like and behaves exactly like the intended object language.

We begin with something like model theory, defining new data types which model the kinds of things that the new language is to be about. The constructs in the language are then given definitions in terms of these new data types. By deduction within HOL we are then able to prove results which correspond to results in the intended object language.

There is a technical term for this kind of treatment of languages in HOL, they are called *shallow semantic embeddings*, and this term indicates that the expressions of the target language are represented by expressions in HOL which are syntactically similar (though perhaps not identical) to those of the intended object language, and which do have the same meaning as the target language expressions. For a fuller description of this kind of method (used in theoretical computer science) see [**?**].

If you have not come across this kind of thing before this probably does not make much sense at this point, but I hope that eventually the material which follows will provide an intelligible example of this method.

Schemas and Higher Order Quantification

Much of the semi-formal material which we are trying to fully formalise involves general talk about the kinds of things which are found in categories. Possibly the formulae are intended as schemas in a first order predicate calculus. This is not the way we will treat them, so a few words explaining why not are in order here.

We are working here in a higher-order logic. In a first-order logic, it is not possible for quantify over anything but individuals. In first order set theory we get around that restriction by having "individuals" which are surrogates for all kinds of higher order objects. In set theory we can, by quantifying over the individuals encompass objects which represent properties of functions of every conceivable type. Some pragmatic issues remain which we need not go into here.

When a first-order formalisation is attempted without benefit of the machinery of set theory, it often proves necessary to use schemata, which are a syntactic surrogate for quantification over higher types. A well known example is the theory PA, a first order version of Peano's axioms for arithmetic. Peano himself formulated his axioms for arithmetic before first order logic was invented, before indeed the foundational problems which provoked the development of type theories. His axiom of induction involved quantification over properties along the following lines:

$$\vdash \forall p\bullet\ p(0) \wedge (\forall x\bullet\ p(x) \Rightarrow p(x + 1)) \Rightarrow \forall x\bullet\ p(x)$$

Which we may paraphrase:

> for all properties p, if p holds for 0 and, whenever p is true of some natural number, it is true also of its successor, then p will be true of all natural numbers

In the first order formalisation of Peano Arithmetic, known as PA, we cannot quantify over properties, so we use instead an axiom schemata, which lifts the quantification into the metalanguage and changes from quantifying over numbers to quantifying over formulae. Thus we have instead something like:

$$\vdash P(0) \land (\forall x \bullet P(x) \Rightarrow P(x + 1)) \Rightarrow \forall x \bullet P(x)$$

Where P is not a predicate in the object language, but a syntactic function in the metalanguage which yields formula. This first order axiom schema describes an infinite set of properly first order axioms obtained by substituting arbitrary formulae (in which 0 occurs) for $P(0)$, and corresponding formulae for $P(x)$ and $P(x+1)$ in which x and $x+1$ respectively replace occurrences of 0 in the original formula.

Features of The Language

The following features of the language are methodologically significant:

[to be supplied]

1.5 Do It Yourself Guide

1.5.1 The Software

1.5.2 ProofPower

The tool used for preparation of this document, for checking the syntax and type correctness of the formal specifications, for assisting in the construction of formal proofs, and for checking the resulting proofs in detail is ProofPower.

This document is a *literate script*. This means that it is a *script* intended for processing by machine, which is also intended to be humanly intelligible (i.e. literate).

The source for the document is machine processed in two distinct ways. The formal content is extracted and processed by the proof tool ProofPower, which understands two main languages, the first, HOL, a kind of higher order logic suitable for the formal development of mathematics and for applications of formalised logic and/or mathematics. The second is a functional programming language called SML, in which instructions may be given to ProofPoweron how to process the formal specifications. This includes instructions on how to construct and check formal proofs of conjectures in HOL.

ProofPowercome with a library of already defined mathematical concepts and of theorems proven in the context of these definitions, which are organised into a hierarchy of theories in which a theory may

make use of the definitions made and theorems proven in any of its ancestors. As ProofPowerprocesses a document it augments the theory hierarchy with the new material. Listings of the theories can be obtained for inclusion at the end of the document.

The other way of processing the source is for the purpose of obtaining a humanly readable document, typically in PDF format, of which this is an example. While the document is being written, the author enters into an interactive dialogue with the proof tool in which new definitions or modifications to existing definitions are checked for grammatical correctness and type-correctness. Proofs of conjecture are developed interactively in such a session using a "goal package" which permits the user to work backwards from the goal he is attempting to prove. The end result of such proof development is a script in the metalanguage SML which provides to ProofPowera prescription for constructing a proof, which is checked for correctness on-the-fly. This will be rerun when the complete document is later processed in batch.

Document preparation uses the LaTeXpackage augmented by facilities provided by ProofPower, various aspects such as the formatting of formal text, the creation of contents lists, indexes and bibliographies being thereby facilitated.

1.5.3 Documentation

1.5.4 The Scripts

Chapter 2

Metaphysical Positivism

2.1 Logical Truth

To model the fundamental notion of "logical truth", we consider various ways in which the semantics of languages can be formally defined. There are many ways of doing this. The choice of how to render the semantics may be part of the process of defining a language or class of languages. Definitions of logical truth are then specific to classes of languages which have the same kind of formal semantics.

2.1.1 Bare-Boned Truth-Conditions

2.1.2 Types

The following "primitive" types are introduced:

SML

$new\_type("S",0);\ \ (*\ sentences\ *)$
$new\_type("C",0);\ \ (*\ contexts\ *)$
$new\_type("W",0);\ \ (*\ possible\ worlds\ *)$
$new\_type("P",0);\ \ (*\ propositions\ *)$

2.1.3 The Semantics

The semantics of our language comes in two parts. A semantic map which delivers propositions, the meanings of sentences in context, and a propositional evaluation map, which extracts truth conditions from a proposition.

The purpose of this document is not to consider the semantics of any particular language but to reason about semantics and concepts defined in terms of semantics. We could have used variables for the semantics, but this time I decided to use loosely defined constants. So the following definitions only tell you the type of the semantic map and the evaluation map, they don't tell you anything more than that, so any results we subsequently obtain using these definitions will hold good for any language with a semantics of the type stipluated here.

HOL Constant

$$\boldsymbol{sm} : S \times C \to P$$

$$T$$

HOL Constant

$$\boldsymbol{pem} : P \times W \to TTV$$

$$T$$

Note that TTV is a type consisting of three 'truth' values, whose names are: $pTrue$, $pFalse$ and pU.

2.1.4 Necessity

A proposition is 'necessarily t' if it takes truth value 't' in every possible world.

HOL Constant

$$\boldsymbol{necessarily} : TTV \to P \to BOOL$$

$$\forall t{:}TTV;\ p{:}P\bullet\ necessarily\ t\ p \Leftrightarrow \forall w{:}W\bullet\ pem(p,\ w) = t$$

A proposition is *necessary* (simpliciter) if it is *necessarily t* for some truth value *t*.

HOL Constant

$$\boldsymbol{necessary} : P \to BOOL$$

$$\forall p{:}P\bullet\ necessary\ p \Leftrightarrow \exists t\bullet\ necessarily\ t\ p$$

HOL Constant

$$\boldsymbol{contingent} : P \to BOOL$$

$$\forall p{:}P\bullet\ contingent\ p \Leftrightarrow \exists w1\ \ w2\bullet \neg\ pem(p,\ w1) = pem(p,\ w2)$$

2.1.5 True in Virtue of Meaning

A common definition of "analytic" is as 'true in virtue of meaning', so we will now try to formalise that idea. If the truth value of a sentence can be ascertained from its meaning only, i.e. without taking into account any 'extra-linguistic fact' (in Quine's words), i.e. without knowing anything about what possible world is actual. This can only be known if it takes the same truth value in every possible world.

This can be generalised to an arbitrary truth value.

Therefore we define:

SML

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

HOL Constant

$$\$\boldsymbol{by\_meaning} : TTV \to (S \times C) \to BOOL$$

$$\forall t{:}TTV;\ s{:}S;\ c{:}C\bullet\ t\ by\_meaning\ (s,\ c) \Leftrightarrow \forall p\bullet\ pem\ (sm(s,\ c),\ p) = t$$

2.1.6 Expresses a Necessary Proposition

My preferred definition of analyticity is that a sentence is analytic if the proposition it expresses is necessarily true. Again we generalise to an arbitrary truth value. I'll make this infix as well.

SML

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

HOL Constant

```
$analytic : TTV → (S × C) → BOOL
```

```
∀t:TTV; s:S; c:C• t analytic (s, c) ⇔ necessarily t (sm(s, c))
```

Now we prove that these two conception of analyticity are the same.

The proof is trivial, exanding the relevant definitions yields a universally quantified identity equation (apart from the names of the bound variables). In the following proof script, the necessary rewriting is broken into two stages to show the identity.

SML

```
set_goal([], ⌜∀t s c• t analytic (s,c) ⇔ t by_meaning (s, c)⌝);
a (pure_rewrite_tac (map get_spec [⌜$analytic⌝, ⌜$by_meaning⌝, ⌜necessarily⌝]));
```

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

(* ?⊢ *)  ⌜∀ t s c• (∀ w• pem (sm (s, c), w) = t) = (∀ p• pem (sm (s, c), p) = t)⌝
```

SML

```
a (rewrite_tac[]);
val analyticity_lemma1 = save_pop_thm "analyticity_lemma1";
```

2.2 KANT'S DEFINITION OF ANALYTICITY

2.2.1 First Shot

Kant defined analyticity only for "subject predicate" sentences, and some have therefore supposed this to be less general than more recent formulations. However, asssuming only that the notion of analyticity is to be preserved by logical equivalence we can show that Kant's definition is equivalent to the preceding ones.

There is an awkwardness in generalising this notion to three truth values, so I will do it only for the one.

We will first define predicate inclusion.

SML
```
declare_infix (300, "contains");
```

HOL Constant
```
$contains : ('a → BOOL) → ('a → BOOL) → BOOL
------------------------------------------------------------
∀P Q• P contains Q ⇔ ∀x• Q x ⇒ P x
```

P *contains* Q is a way of writing a subject predicate assertion in which the subject is Q and the predicate is P (this is Aristotelian terminology, we don't use predication in this way in modern logic).

Now we show that every judgement is equivalent to one in "subject predicate" form:

```
kantian_lemma =
      ⊢ ∀ SS• ∃ P Q• SS ⇔ P contains Q
```

SML
```
set_goal([], ⌜∀SS• ∃P Q• SS ⇔ P contains Q⌝);
a (strip_tac THEN ∃_tac ⌜λx• SS⌝ THEN ∃_tac ⌜λx• T⌝);
a (rewrite_tac [get_spec ⌜$contains⌝]);
val kantian_lemma = save_pop_thm "kantian_lemma";
```

This lemma may be applied generally, thus:

SML
```
val ℕ_gt_trans = ∀_elim ⌜∀x y z:ℕ• x > y ∧ y > z ⇒ x > z⌝ kantian_lemma;
```

yields:

```
val ℕ_gt_trans =
   ⊢ ∃ P Q• (∀ x y z• x > y ∧ y > z ⇒ x > z) ⇔ P contains Q
```

However, this is smoke and mirrors, because though it appears to be saying something about propositions, it is really about truth values. i.e. we have proven that every sentence has the same truth value as some sentence in subject predicate form, i.e. that there is a false sentence and a true sentence in subject predicate form.

To have a relevance to the scope of Kant's definition of analyticity we need some real metatheoretic reasoning in which we talk about meanings of judgements.

2.2.2 Fuller Treatment

The treatment in the previous section is not wholly convincing.

A sufficient reason for this is that the central thesis is not itself formalised. Insofar as there is any doubt about the thesis it therefore fails to improve the situation, and the question arises whether the theorems proven really establish the intended result.

The thesis is that, under certain provisos, Kant's definition of analyticity is equivalent to a definition along the lines of "true in virtue of meaning". However, it seems probable that the conditions under which Kant's definition holds good differ significantly from those in which the other definition is applicable. To formulate the thesis it is therefore necessary to establish some sufficient (and preferably necessary) conditions for both definitions to be applicable.

The required result is a general result covering a class of descriptive languages, and I think this will be better expressed if we abandon the previous methods of formalising in relation to some loosely specified but fixed (constant) language, and talk about languages using variables.

I will therefore start from scratch but replicate the same basic idea of what a descriptive language is.

Kant talks about "judgements", I will treat these as sentences in context.

2.2.3 Types

In this version type variables will be used where constants were previously used, as follows:

'S Sentences

'C Contexts

'P Propositions

'W Possible Worlds

2.2.4 The Semantics

The semantics of a 'descriptive' language comes in two parts. A semantic map which delivers propositions, the meanings of sentences in context, and a propositional evaluation map, which extracts truth conditions from a proposition.

The purpose of this document is not to consider the semantics of any particular language but to reason about semantics and concepts defined in terms of semantics.

The following type abbreviations give the type of the semantics of a descriptive language.

SML

```
(* Semantic Map *)
declare_type_abbrev ("SM", ["'C", "'P", "'S"], ⌜:'S × 'C → 'P⌝);

(* Propositional Evaluation Map *)
declare_type_abbrev ("PEM", ["'P", "'W"], ⌜:'P × 'W → BOOL⌝);

(* Language *)
declare_type_abbrev ("LAN", ["'C", "'P", "'S", "'W"], ⌜:('C, 'P, 'W)SM × ('P, 'W)PEM⌝);
```

2.2.5 Necessity

A proposition is 'necessary' if it takes truth value 'T' in every possible world, the definition is parameterised by a propositional evaluation map.

HOL Constant

$$\textbf{\textit{necessary}} : ('P, 'W)PEM \to 'P \to BOOL$$

$$\forall pem\ (p{:}'P)\bullet\ necessary\ pem\ p \Leftrightarrow \forall w{:}'W\bullet\ pem(p,\ w) = T$$

HOL Constant

$$\textbf{\textit{contradictory}} : ('P, 'W)PEM \to 'P \to BOOL$$

$$\forall pem\ (p{:}'P)\bullet\ contradictory\ pem\ p \Leftrightarrow \forall w{:}'W\bullet\ pem(p,\ w) = F$$

HOL Constant

$$\textbf{\textit{contingent}} : ('P, 'W)PEM \to 'P \to BOOL$$

$$\forall pem\ (p{:}'P)\bullet\ contingent\ pem\ p \Leftrightarrow \exists w1\ w2{:}'W\bullet \neg\ pem(p,\ w1) = pem(p,\ w2)$$

[to be completed!]

Chapter 3

Conclusions

Index