Abstract

This is a continuation of a paper in which the role of types in semantic theory was explored. Inevitably, in that paper some issues in type theory and semantics were treated only in passing. In particular, little was said about Stratified Type Theories. This paper concentrates on these theories and provides a theoretical framework for semantic applications.

Keywords: Semantics, types, type theory, stratification, intensionality.

1 Why stratify?

Our objective in this paper is to develop a class of stratified theories aimed at the foundations of natural language semantics. Our motivation stems from the fact that natural language seems to require quantification over propositions, and such quantification introduces, ipso facto, a form of self-reference. Unfortunately, this leads to an empirical form of the liar paradox [18]. To see this let $p$ be the predicate be said by a Cretan. Now consider the following sentence expressed in Simple Type Theory (STT) or Higher Order Logic (HOL) — where $P$ is the type of propositions:

$$\forall x : P \cdot p(x) \rightarrow \neg x.$$  \hfill (1)

It is a matter of fact that Epimenides was a Cretan and he said this, i.e.

$$p(\forall x : P \cdot p(x) \rightarrow \neg x).$$  \hfill (2)

In STT/HOL, (1) and (2) entail that

$$\exists x : P \cdot p(x) \land x.$$  \hfill (3)

But be aware that this derivation requires (1) itself to be the witness in (3). Now suppose that Cretans say very little; indeed, suppose the only thing a Cretan could ever have said is (1), i.e.

$$\forall x : P \cdot p(x) \rightarrow x = (\forall x : P \cdot p(x) \rightarrow \neg x).$$  \hfill (4)

Now we can very easily derive a contradiction. However, it is not one that involves set membership or truth, but rather one that stems from indirect discourse involving say.

Thomason [18], argues that stratification provides a principled way of blocking these paradoxes, since stratification provides a principled way of stopping sentence (1) from coming within the scope of its bound variable. Thomason further argues that stratified theories \(^1\) are naturally intensional, i.e. they yield a fine-grained notion of proposition. This follows from the observation that, on a stratified view, the type of propositions is created in layers. As a consequence, in sentence (1), the type $P$ cannot be the totality of propositions. However, if we identify propositions with truth values or even

\(^1\)Thomason uses the more traditional term Ramified Type Theory, but the difference is cosmetic.
sets of possible worlds, then it is hard to maintain the stratified position; if we have such a totality at our disposal the motivation, to see them constructed layer by layer, is lost. Consequently, we require a notion of proposition that is even more fine-grained than that returned by the possible world account. Certainly, we will not be able to endorse a theory where any form of logical equivalence determines propositional identity. These seem like sufficient reasons to explore stratified theories.

In the course of the paper we shall provide a stock of such theories. These theories will subsume both the Ramified Theories of Types [18, 8, 9] and Constructive Type Theory (CTT). However, initially, we develop some base theories which will, as the name suggests, form a base for our stratified ones.

2 A base grammar (G₀)

A traditional problem with stratified theories is their form: they are hard to formulate in an elegant syntactic way. In the standard approach [4, 18] the expressions of the language become overburdened with type information and, consequently, the usefulness of the theory is undermined. We shall attack this problem via an approach to typing which has its roots in the type inference systems of the Lambda Calculus. More explicitly, we develop the syntax of our language as a deductive system whose theorems determine the syntactic categories and their contents. This will yield a generalized notion of grammar that will provide the machinery for the development of more tractable stratified theories.

The grammatical system has three grammatical judgements:

\[ s \text{ type} \]

\[ t \text{ prop} \]

\[ t : s. \]

The first states that \( s \) is a type term, the second that \( t \) is a propositional term and the last that \( t \) has type \( s \). To make matters more palatable, we shall often use upper case roman letters for type terms and lower case Greek letters for propositional terms. However, this convention has no formal force; it is only a presentational aid. We shall use \( \Theta \) for a judgement of any of the above kinds. These judgements take place with respect to a set of type assignments to variables, i.e. a context \( c \) of the form

\[ x_1 : T_1, \ldots, x_n : T_n. \]

2 A more implicit theme of the paper relates to it being a continuation of [21]. Since it was written there has been some significant activity in type theory and semantics and we shall use the cloak of stratification to point out connections with some of the recent work and, in particular [5, 7, 22].

3 These formulations include a type checking inference for a polymorphic version of Ramified type theory. They have some connection with the theories that we shall later produce, although the exact connections has to be a topic of further investigation.

4 The latter of course, is already in employment in semantic theory [3, 14, 5].

5 The terms of the language are selected from those given by the following BNF syntax.

\[ t ::= x | \lambda x : T \cdot t | tt | (t, t) | \pi_1(t) | \pi_2(t) \]

\[ I | t \parallel t | t \Rightarrow t \]

\[ t = T | t \in \Omega | t \land t | t \lor t | t \to t \]

\[ \forall x : t \cdot t | \exists x : t \cdot t. \]

Our rules then carve out the truly grammatical subsets.
We shall call the theory $G_0$ and write
\[ c \vdash_{G_0} \Theta \]
(dropping the suffix where convenient) for the sequent which asserts that $\Theta$ follows from the context $c$.

$G_0$ is determined by the following sequent style rules. We shall only indicate changes in the context where there is a change from premises to conclusion. We begin with the structural rules. The first two are the standard assumption and thinning rule and the last the obvious substitution rule.

\[
\begin{align*}
\frac{c \vdash T \text{ type}}{c, x : T \vdash x : T} & \quad \frac{c \vdash T \text{ type} \quad c \vdash \Theta}{c, x : T \vdash \Theta} \\
\frac{c, x : T \vdash \Theta[x]}{c : T \vdash \Theta} & \quad \frac{c \vdash t : T}{c \vdash t \vdash T}
\end{align*}
\]

The next group of rules govern type formation. The rules inform us that there is a basic type $I$ (of individuals) and that the class of types is closed under the formation of operation/function types and Cartesian products. The ambiguity over the term operation/function alludes to the fact that extensionality for operations is an option. Finally, note that the inclusion of Cartesian products here is a luxury rather than a necessity, but a useful one.

\[
\begin{align*}
\frac{I \text{ type}}{T \text{ type} \quad S \text{ type}} & \quad \frac{T \text{ type} \quad S \text{ type}}{T \Rightarrow S \text{ type} \quad T \text{ type} \quad S \text{ type}}
\end{align*}
\]

This third group of rules govern the second judgement, i.e. being a proposition. The actual rules are as one would expect, but notice that equality bears its type on its sleeve and $\Omega$ denotes absurdity.

\[
\begin{align*}
t : T & \quad s : T \quad \Omega \text{ prop} \\
\phi \text{ prop} & \quad \psi \text{ prop} \quad \phi \text{ prop} \quad \psi \text{ prop} \\
\phi \lor \psi \text{ prop} & \quad \phi \text{ prop} \quad \psi \text{ prop} \quad \phi \text{ prop} \quad \psi \text{ prop}
\end{align*}
\]

Finally, we provide the rules for membership in types. Again, these are standard. The first pair provide the introduction and elimination rules for operations and the second, the rules for pairing and projections.

\[
\begin{align*}
c, x : A & \vdash t : B \\
\frac{c \vdash \lambda x : A \cdot t : A \Rightarrow B}{s : A \Rightarrow B} & \quad \frac{a : A \quad b : B}{(a, b) : A \otimes B} \\
\frac{a : A_1 \otimes A_2}{\pi_i(a) : A_i} & \quad i = 1, 2.
\end{align*}
\]

We employ the standard definitions for negation and logical equivalence. This completes the statement of the grammar. We shall now spend a little time exploring it. Our first result informs us that contexts in the theory are sensible and that the judgement of being a type is context independent — although not a big surprise here, in later systems it will fail.
PROPOSITION 2.1

(i) If \( c \vdash T \) type then \( \vdash T \) type, i.e., from an empty \( c \).
(ii) \( c \vdash T \otimes S \) type iff \( c \vdash T \) type and \( c \vdash S \) type.
(iii) \( c \vdash T \Rightarrow S \) type iff \( c \vdash T \) type and \( c \vdash S \) type.
(iv) If \( c \vdash \Theta \), then for each \( t \) that occurs in a type assignment \( x : t \) in \( c \),
\( c \vdash t \) type.
(v) If \( c \vdash t : s \) type then \( c \vdash s \) type.

PROOF. By simultaneous induction on the structure of derivations. The details are tedious but straightforward.

PROPOSITION 2.2

(i) \( c \vdash t =_T s \) prop iff \( c \vdash t : T \) and \( c \vdash s : T \).
(ii) \( c \vdash \phi \circ \psi \) prop iff \( c \vdash \phi \) prop and \( c \vdash \psi \) prop, \( \circ = \vee, \wedge, \rightarrow \).
(iii) \( c \vdash \forall x : T \cdot \phi \) prop iff \( c, x : T \vdash \phi \) prop.
(iv) \( c \vdash \exists x : T \cdot \phi \) prop iff \( c, x : T \vdash \phi \) prop.
(v) \( c \vdash st : S \) iff \( c \vdash s : T \Rightarrow S \) and \( c \vdash t : T \) for some \( T \).
(vi) \( c \vdash \lambda x : T \cdot t : T \Rightarrow S \) iff \( c, x : T \vdash t : S \).
(vii) \( c : (a, b) : A_1 \otimes A_2 \) iff \( c \vdash a : A_1 \) and \( c \vdash b : A_2 \).
(viii) \( c \vdash \pi_1(a) : A_1 \) and \( c \vdash \pi_2(a) : A_2 \) iff \( a : A_1 \otimes A_2 \).

PROOF. By induction on the structure of derivations. The directions from right to left follow immediately from the rules. The other directions are checked using a case analysis on the possible ways the various conclusions can be established. If the conclusion follows from the grammar introduction rules for the connective, the result is immediate. If the conclusion is the result of a structural rule, the result follows from using the structural rule itself. For example, suppose the last step in the derivation is the following instance of an application of the second structural rule.

\[
\begin{align*}
\frac{c \vdash T \text{ type} \quad c \vdash \phi \wedge \psi \text{ prop}}{c, x : T \vdash \phi \wedge \psi \text{ prop}}
\end{align*}
\]

Consider the premisses. By induction, we may suppose that
\( c \vdash \phi \) prop and \( c \vdash \psi \) prop.

By the structural rule itself we have the following:
\( c, x : T \vdash \phi \) prop and \( c, x : T \vdash \psi \) prop.

We now use the conjunction rule.

These results shows us how to construct a syntactic analyser. Indeed, they justify calling \( G_D \) a grammar. Our next result shows us that, in this system at least, we have unicity of types. This will not be quite true in all the later extensions.

PROPOSITION 2.3 (Unicity of Types)

If \( c \vdash t : T \) and \( c \vdash t : S \) then \( S \) and \( T \) are identical.

PROOF. By induction on the derivations. All the steps are routine. For example, suppose that both
\[
\begin{align*}
c &\vdash \lambda x : A \cdot t : A \Rightarrow B \\
c &\vdash \lambda x : A \cdot t : A \Rightarrow C 
\end{align*}
\]
Then we carry out a case analysis on how this was deduced. For example, suppose both are the result of introduction rule.

\[
\begin{align*}
\frac{c, x : A \vdash t : B}{c \vdash \lambda x : A \cdot t : A \Rightarrow B} & \quad \frac{c, x : A \vdash t : C}{c \vdash \lambda x : A \cdot t : A \Rightarrow C}
\end{align*}
\]

Then by induction we must have \( B = C \).

Finally, we have the following.

**Proposition 2.4 (Substitution)**

Suppose that \( c \vdash s : S \) then

1. If \( c \vdash \phi[s/x] \text{ prop} \) then \( c, x : S \vdash \phi[x] \text{ prop} \).
2. If \( c \vdash t[s/x] : T \) then \( c, x : S \vdash t[x] : T \).

**Proof.** By simultaneous induction on the derivations. For example, consider the rule

\[
\frac{a[s/x] : T}{a[s/x] = T} \quad \frac{b[s/x] : T}{b[s/x] \text{ prop}}
\]

By induction, via 2, we have

\[
c, x : S \vdash a[x] : T \text{ and } c, x : S \vdash b[x] : T.
\]

By the rule itself we are finished.

This is a rather different approach to syntax than that delivered by a standard context free grammar/BNF definition. \( G_0 \) is best seen as a Curry version of the Church style theory with grammar rules such as

\[
x^T \text{ is a term of type } T.
\]

If \( t \) is a term of type \( S \) then \( \lambda x^T \cdot t \) is a term of type \( T \Rightarrow S \).

In the Curry approach, the grammar rules must be given relative to a context, but relative to such a context, they inform us how to construct elements of the various types — and there are no terms except those of some type. Such an approach generalizes BNF and, as we shall see, facilitates a smooth expression of the main stratified theories.

**3 A base logic \( (C_0) \)**

The base logic uses the grammar but goes beyond it to support full logical inference. Judgements include the grammatical ones of \( G_0 \), together with the assertion of truth

\[
\phi \text{ true}
\]

that we shall usually write without the true decoration. A context \( \Gamma \) is now a set of assumptions

\[
\phi_1, \ldots, \phi_n
\]

of two possible kinds:

\[
x : S
\]

\[
\phi \text{ true}
\]
i.e. a type assignment or the assertion that something, hopefully a proposition, is true. Some notation will facilitate the statement of the rules. We shall use $c$ to denote the subset of type assignments of $\Gamma$. We shall also, where the context is clear, just use $c$ for grammatical contexts. $\Gamma[s/x]$ denotes $\Phi_{\Gamma}[s/x]$, i.e. substitution in the propositional terms of $\Gamma$.

We write

$$\Gamma \vdash _c \Theta$$

(or without the subscript where it is clear which system is being used) if the judgement $\Theta$ (of any of the above four kinds) follows from assumptions $\Gamma$ by the following rules. We divide the rules into several classes. We assume the rules of $G_0$ for the grammatical judgements. In addition, we have the following new ones which govern the assertion of truth. First, the logical structural rules follow the pattern of the grammatical ones. However, the substitution rule for the new judgement is derivable from the universal quantifier rules, so we don’t need to state it. Hence, by way of structural rules, all we need, in addition to those of $G_0$, are the following:

$$\frac{c \vdash _G \phi \text{ prop}}{c, \phi \vdash \phi} \quad \frac{\Gamma \vdash \Theta \quad c \vdash _G \phi \text{ prop}}{\Gamma, \phi \vdash \Theta}$$

We require the standard equality rules for a typed theory.

$$\frac{c \vdash _G t : T}{c \vdash t =_T t} \quad \frac{t =_T s}{\phi[t]}$$

Along with these go the special equality rules for our various term constructs.

$$\frac{x : A \vdash t : B \quad s : A}{(\lambda x : A \cdot t)s =_B t[s/x]} \quad \frac{(t_1, t_2) : A_1 \otimes A_2}{\pi_i(t_1, t_2) =_{A_i} t_i \quad i = 1, 2} \quad \frac{t : A_1 \otimes A_2}{t =_{A_1 \otimes A_2} (\pi_1(t_1, t_2), \pi_2(t_1, t_2))}$$

Finally, we provide the logical rules. As usual, for each logical constant, there are rules of introduction and elimination.

$$\frac{\Gamma \vdash \Omega \quad c \vdash _G \phi \text{ prop}}{\Gamma, \neg \phi \vdash \Omega} \quad \frac{\Gamma \vdash \neg \phi \quad c \vdash _G \psi \text{ prop}}{\Gamma, \phi \vdash \psi} \quad \frac{\Gamma, \neg \phi \vdash \psi}{\Gamma \vdash \phi \land \psi} \quad \frac{\Gamma \vdash \phi \land \psi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi}$$

$$\frac{\Gamma \vdash \phi \quad c \vdash _G \psi \text{ prop}}{\Gamma \vdash \phi \lor \psi} \quad \frac{\Gamma \vdash \psi \quad c \vdash _G \phi \text{ prop}}{\Gamma \vdash \phi \lor \psi} \quad \frac{\Gamma, \phi \vdash \eta \quad \Gamma, \psi \vdash \eta}{\Gamma, \phi \lor \psi \vdash \eta}$$

$$\frac{\Gamma, \phi \vdash \psi \quad \Gamma \vdash \phi \to \psi}{\Gamma \vdash \phi \to \psi}$$

$$\frac{\Gamma, x : T \vdash \phi}{\Gamma \vdash \forall x : T \cdot \phi} \quad \frac{\Gamma \vdash \forall x : T \cdot \phi \quad c \vdash _G t : T}{\Gamma \vdash \phi[t/x]} \quad \frac{c \vdash _G t : T \quad \Gamma \vdash \phi[t/x]}{\Gamma \vdash \exists x : T \cdot \phi} \quad \frac{\Gamma \vdash \exists x : T \cdot \phi \quad \Gamma, x : T, \phi \vdash \eta}{\Gamma \vdash \phi \vdash \eta}$$
We assume the normal side conditions on the quantifier rules. This completes the basic logic. The rules are relatively standard ones for a classical typed predicate logic, with the extra conditions which ensure that nothing which is not provably a proposition, is provable. We shall call this the Coherence property of the logic. This seems essential: the grammar picks out the intended meaningful strings and we do not wish meaningless strings to be provable in the logic.

**Theorem 3.1 (Coherence theorem)**

If \( \Gamma \vdash \phi \) then \( c_\Gamma \vdash_{G_\alpha} \psi \text{ prop} \) for each \( \psi \text{ in } \{ \phi \} \cup \Phi_\Gamma \).

**Proof.** By induction on the structure of derivations. We first consider the inference rules themselves. Most of the rules are easy to check; we provide some illustrations. Consider first the implication introduction rule. Consider the premiss

\[
\phi \vdash \psi.
\]

By induction, \( \phi \text{ prop} \). Also by induction, \( \psi \text{ prop} \). This yields \( \phi \rightarrow \psi \text{ prop} \). Next consider disjunction introduction. The premiss is

\[
\vdash \phi \quad \vdash \psi \text{ prop}.
\]

By induction, \( \phi \text{ prop} \). The result now follows from the formation rule for disjunction. The elimination rule is immediate. Next consider existential quantification introduction.

\[
\frac{c \vdash s : T \quad c \vdash \phi[s/x]}{c \vdash \exists x : T \cdot \phi}
\]

Given the induction hypothesis for the premises, \( c, x : T \vdash \phi \text{ prop} \) follows from the substitution proposition. The result now follows from the formation rule for the existential. Now consider the structural rules. The first two are automatic. For substitution, we illustrate with the following case, where we wish to show that \( \phi \) is a proposition:

\[
\frac{\Gamma, \phi, x : T \vdash \Theta \quad \Gamma \vdash s : T}{\Gamma[s/x], \phi[s/x] \vdash \Theta[s/x]}
\]

By induction, it is sufficient to observe that the following is itself an instance of substitution:

\[
\frac{c_\Gamma, x : T \vdash \phi \text{ prop} \quad c_\Gamma \vdash_{G_\alpha} s : T}{c_\Gamma \vdash \phi[s/x] \text{ prop}}
\]

Finally, we observe that the theories of type membership in the grammar and the logic agree. This will not always be so. Indeed, coherence always guarantees that the logic is in harmony with the grammar. However, the other direction depends upon the simple type structure of \( G_\alpha \).

**Proposition 3.2**

\[
c \vdash_{G_\alpha} t : T \text{ iff } c \vdash_{c_\alpha} t : T.
\]

**Proof.** The right to left direction follows from the coherence property. The other follows because the grammar is a subsystem of the logic.
This is the classical theory. However, if we drop the rule
\[ \Gamma, \neg \phi \vdash \Omega \]
\[ \Gamma \vdash \phi \]
we get the Intuitionistic version, that we shall call \( \text{I}_0 \). These intuitionistic theories, are related to intuitionistic arithmetic in higher types. More precisely, if we add the axioms for arithmetic, the intuitionistic versions of the base theories are Curry style versions of HA\(^\omega\) [16]. Such theories are conservative over first order intuitionistic arithmetic.

### 4 Stratified theories

This theory is not expressive enough for semantic purposes and the reason is simple: we need a type of propositions. For example, for a compositional semantics on a Montague style, we need to be able to represent quantifiers such as:
\[ \text{Every} = \lambda f : I \Rightarrow P \cdot \lambda g : I \Rightarrow P \cdot \forall x : I \cdot f \cdot x \rightarrow gx \]
\[ \text{Some} = \lambda f : I \Rightarrow P \cdot \lambda g : I \Rightarrow P \cdot \exists x : I \cdot f \cdot x \land gx \]
and for this we need a type (\( P \)) of propositions. Even if we take an approach to determiners/quantifiers where they are added as primitives, the type propositions, on at least one influential analysis, will still be needed to deal with indirect discourse and belief statements.

To this end we now add a new type that we shall call, for reasons soon to be made clear, \( \text{G}_0 \). Its members are the propositions of \( \text{G}_0 \). More precisely, let \( \text{G}_1 \) be the new grammar whose rules are those of \( \text{G}_0 \), but extended by addition of the following rules:

\[ \vdash_{\text{G}_1} P_0 \text{ type} \quad \vdash_{\text{G}_1} \phi \text{ prop} \quad x : P_0 \vdash_{\text{G}_1} x \text{ prop} \]

Observe that, the last two yield the following:
\[ \text{If } c \vdash_{\text{G}_0} \phi \text{ prop then } c \vdash_{\text{G}_1} \phi \text{ prop}. \]

But \( \text{G}_1 \) lets more in because of the enriched class of types. The above rules yield the closure conditions of the first universe of CTT. For symmetric elegance, and indeed later purposes, we also add a second new type whose elements are the types of \( \text{G}_0 \). This is governed by the following rules that parallel those for the propositions of \( \text{G}_0 \):

\[ \vdash_{\text{G}_1} U_0 \text{ type} \quad c \vdash_{\text{G}_0} T \text{ type} \quad x : U_0 \vdash_{\text{G}_1} x \text{ type} \]

We then get the new logical theory \( \text{C}_1 \) by extending all the rules of \( \text{C}_0 \) to this new language, i.e. replacing \( \text{G}_0 \) by \( \text{G}_1 \) in the rules for \( \text{C}_0 \).

We can then generalize to obtain a hierarchy of theories. Let \( \text{G}_n / \text{C}_n \) be given. We obtain \( \text{G}_{n+1} / \text{C}_{n+1} \) from \( \text{G}_n \) in a parallel fashion to the above, i.e. we add new types of propositions \( \text{P}_n \) and types.

---

6It is worth pointing out that in \( \text{G}_0 \), we have the rudiments of a functional programming language together with some additional logical apparatus. In this regard, semantics based upon such a system would be in keeping with the recent work of [22] to base semantics on the type structure of functional languages such as Haskell [10]. We shall see this more clearly later.
brought into being by the following rules:

\[
\begin{align*}
\vdash_{\mathcal{G}_{n+1}} P_n \text{ type} & \quad \vdash_{\mathcal{G}_n} \phi \text{ prop} \\
\vdash_{\mathcal{G}_{n+1}} P_n \text{ type} & \quad \vdash_{\mathcal{G}_n} \phi : P_n \\
x : P_n \vdash_{\mathcal{G}_{n+1}} x \text{ prop} & \quad \vdash_{\mathcal{G}_n} T \text{ type} \\
\vdash_{\mathcal{G}_{n+1}} U_n \text{ type} & \quad \vdash_{\mathcal{G}_n} T : U_n \\
x : U_n \vdash_{\mathcal{G}_{n+1}} x \text{ type}
\end{align*}
\]

The new logic \( \mathcal{C}_{n+1} \) is then obtained by extending all the rules to this new grammar. The Intuitionistic systems \((\mathcal{I}_n, n \geq 0)\) are obtained, as before, by dropping the classical negation rule. Observe that we no longer have a simple unicity of types property, since now we have accumulation, i.e.

\[
\begin{align*}
\text{If } c & \vdash_{\mathcal{G}_{n+1}} \phi : P_n \text{ then } c \vdash_{\mathcal{G}_{n+2}} \phi : P_{n+1}.
\end{align*}
\]

However, every expression that has a type in the hierarchy, has a minimal one.

Let \( \mathcal{C} \) be the theory that is the union of all the logics \( \mathcal{C}_n, n \geq 0 \). This theory differs from that proposed in [18] in two ways. First, the latter is based on a formulation where the syntax is given via a traditional two-level grammar and the variables and terms are decorated with type information. Secondly, the types are different; [18] is based upon STT and ours on HOL — both stratified of course. Indeed, if we strip away the operation types and Cartesian product types and replace them with types of the following form

\[
(\mathcal{T}_1, \ldots, \mathcal{T}_n / P_m)
\]

we obtain the Ramified theory of Thomason [18]. Of course, these types are available to us. Thus our layers are similar in content and motivation to those of Russell’s original stratified type theory. However, our theory will enable the generalization to dependent types, a topic we shall pursue later.

Despite the fact that no one has complained that stratification cripples or even hinders the semantic enterprise, one question we should address is: are stratified theories expressive enough for it? In terms of bread and butter semantics, it matters not whether the theory is stratified; in either case, semantics proceeds in the same way; one is hardly aware that the whole theory is stratified.

The only issue is whether natural language semantics has also to provide a semantics for the whole of the language of mathematics, and in particular, set theory. We suggest not. Our foundational objective is to describe the semantic mechanisms of natural language and the theory should, in Feferman’s terms, be Faithful and Adequate. If set theory is to be taken as part of natural language, it is a highly specialized part and to let it dictate the whole of the semantic enterprise, distorts matters and we lose the goal of the foundational enterprise. Any fool can do semantics in set theory, higher order logic or the theory of constructions.

5 Intensionality

We have already argued that stratified systems naturally lead to a fine-grained notion of proposition. The converse, namely that some form of intensionality is required for semantics, is now pretty much accepted. We now examine what this means for our theories. To begin with consider the following Axiom of Extensionality for propositions:

\[
\forall x : P_n \cdot \forall y : P_n \cdot (x \leftrightarrow y) \rightarrow x =_{P_n} y,
\]

i.e. any two propositions that are provably equivalent, have the same content. Should we adopt it? In the current logical setting this would certainly lead to a notion of proposition that is too
coarse grained. After all we do not even have possible worlds to soften the force of this axiom of extensionality. However, consider the following extensionality axiom — this time for operations:

\[ \forall f, g : S \Rightarrow T \cdot (\forall x : S \cdot f x =_T g x) \Rightarrow f =_S T \cdot g. \] (EXT\(_{OP}\))

If we had this we could interpret operations as functions. Is this also to be excluded? Aczel [2] argues not. He argues that, if we reject EXT\(_P\), we do not also need to reject EXT\(_{OP}\). In particular, consider the following instance:

\[ \forall f, g : S \Rightarrow P_n \cdot (\forall x : S \cdot f x =_{P_n} g x) \Rightarrow f =_{S=_{P_n}} g. \]

Given that the notion of propositional equality is more demanding than extensional equivalence, propositional operations inherit this fine-grained notion, i.e. intensionality resides in the propositions and is inherited by the operations.

Although the current theory distinguishes between extension and intension, it only does so implicitly. More precisely, the expressions of the theory can all be taken to be intensional in so far as propositions are not extensionally determined, i.e. EXT\(_P\) is not assumed. Propositions only act extensionally when used in the logical theory as vehicles of reasoning — and this is reflected in the logical theory not the grammar. This seems to be, in spirit at least, close to the approach to semantics in [22].

One can go further and mark the intension/extension distinction more explicitly by adding a new type to the theory, i.e. Bool — the type of truth values. The grammar \(G_1\) is then enriched by the rule

\[ \frac{\vdash_{G_1} \phi : P_0}{\vdash_{G_1} T(\phi) : Bool} \]

However, to sensibly complete the syntactic separation of propositions functioning in intension and extension, and the fact that we now have the type of truth values as a first class type in the theory, we may replace the judgement \(\phi \text{ true}\) with \(T(\phi)\). One might go even further and insist [17] on every expression having both an extension and an intension. This would seem necessary on a pure Montague style approach [12, 13].

6 Dependency

We claim that the theoretical set-up encoded in the present theory is a minimal stratified type theory for natural language semantics. However, the recent history of semantics has seen more and more type constructors being employed in an effort to obtain representations of more and more natural language phenomena. Discourse phenomena have occupied central stage here and this has resulted in a greater sensitivity to contextual issues. In turn, this has generated an interest in dependent types, where we shall use this term to mean that the notion of being a type and its membership may depend upon contextual information.

In the current theory there is a disharmony between types and propositions: the latter depend upon contexts but the former do not. A natural way of permitting types to do so is to allow them to depend upon propositions and, a useful way of facilitating this, is to allow separation types. We employ the base theories to illustrate how the extensions are generated. We extend the grammar \(G_0\) by adding a

\[ \frac{\vdash_{G_1} \phi : P_0}{\vdash_{G_1} T(\phi) : Bool} \]
new type constructor given by the following rules.

\[
\begin{align*}
    c, x : T &\vdash \phi \text{ prop} \\
    c &\vdash \{x : T \cdot \phi\} \text{ type} \\
    c, x : T &\vdash \phi \text{ prop} \quad c \vdash t : T \\
    c &\vdash \{x : T \cdot \phi\} \\
\end{align*}
\]

Notice that now we need the context \(c\) to guarantee that something is a type; it is not independent as in the case of the types of \(G_0\). Moreover, types may contain variables and contexts must now be thought of as sequences \(x_1 : T_1, \ldots, x_n : T_n\) where, in the theory, the free variables of \(T_{i+1}\) are a subset of \(\{x_1, \ldots, x_i\}\). Call this system \(G_0 + \text{Sep}\).

With these remarks incorporated, the important elementary propositions which govern \(G_0\) extend to this new grammar. In particular, the latter informs us that we still have a grammar. In this regard, the second rule for separation types does not depend upon the truth of the proposition \(\phi\); only on its grammaticality. But, as we shall see shortly, in the logic, matters are different.

**Proposition 6.1 (Syntactic Analysis)**

\(c \vdash_{G_0 + \text{Sep}} t : \{x : T \cdot \phi\}\) iff \(c \vdash_{G_0 + \text{Sep}} t : T\) and \(c, x : T \vdash \phi\) prop.

**Proof.** It is a simple extension of the original, i.e. subtypes generate a new case. Again, we have only to consider the possible ways in which the left-hand side can arise: either directly or by the application of a structural rule and all these cases are easy to check. \(\blacksquare\)

For the logic we extend this theory with the following rules, where the membership condition in the types is now more demanding.

\[
\begin{align*}
    t : T &\vdash \phi[t/x] \\
    t &\vdash \{x : T \cdot \phi\} \\
    t &\vdash \phi[t/x] \quad t : T \\
    t &\vdash \{x : T \cdot \phi\} \\
\end{align*}
\]

Call this theory \(C_0 + \text{Sep}\). The coherence theorem extends.

**Theorem 6.2 (Coherence theorem (\(C_0 + \text{Sep}\)))**

If \(\Gamma \vdash_{C_0 + \text{Sep}} \phi\) then \(c_{\Gamma} \vdash_{G_0 + \text{sep}} \psi\) prop for each \(\psi\) in \(\{\phi\}\) \(\cup \Phi_{\Gamma}\).

However, type membership in the logic and type membership in the grammar now part company: it is no longer the case that

\(\text{If } c \vdash_{G_0 + \text{Sep}} t : T \text{ then } c \vdash_{C_0 + \text{Sep}} t : T\).

On the other hand, because of coherence, the converse does hold. Although type membership in the grammar is still decidable, it is not so in the logic since the judgement of type membership in the logic is intimately bound up with the judgement of truth itself. This is an important conceptual issue since we wish to maintain the \(G\)-sequence of theories as Grammars.

The corresponding stratified theories (now named \(G_1 + \text{Sep}\)) are obtained as before but now there is interdependence between propositions and types; they both depend upon each other. We then get the new logical theory \(C_1 + \text{Sep}\) by extending all the rules of \(C_1\) to this new setting.

With subtypes in place we may generalize the basic type constructors of \(G_0\). This will move us closer to the type structure of constructive type theories (CTT). This theory (call it \(G_0' + \text{Sep}\)) is obtained from \(G_0 + \text{Sep}\) by replacing the grammatical rules for operations by the following.

\[
\begin{align*}
    x : A &\vdash B \quad \text{type} \\
    \prod x : A \cdot B &\text{ type} \\
    x : A &\vdash B \quad \text{type} \\
    \Sigma x : A \cdot B &\text{ type} \\
\end{align*}
\]
Their introduction and elimination rules are then a simple generalization of operations and products.

\[
\begin{align*}
  x : A & \vdash s : B[x] & s : \Pi x : A \cdot B & \vdash t : A \\
  \lambda x : A \cdot s : \Pi x : A \cdot B & \quad st : B[t/x] \\
  a : A & \vdash b : B[a/x] & a : \Sigma x : A \cdot B & \vdash (a, b) : \Sigma x : A \cdot B \\
  (a, b) : \Sigma x : A \cdot B & \vdash \pi_1(a) : A & \vdash \pi_2(a) : B[\pi_1(a)/x]
\end{align*}
\]

These generalize the rules for the operations, i.e. where \( x \notin FV(B) \) we effectively obtain the types \( A \Rightarrow A \Rightarrow B, A \bowtie B \). However, be aware that they require separation to be in place for them to generalize matters. We obtain the obvious generalizations of all the main propositions of the grammar.

The logical rules are then enriched by the following rules of equality for these dependent products and sums.

\[
\begin{align*}
  x : A & \vdash s : B[x] & t : A \\
  (\lambda x : A \cdot s)t =_{B[t/x]} & \vdash st : B[t/x] \\
  (a, b) : \Sigma x : A \cdot B & \vdash \pi_1(a, b) =_{A \wedge \pi_2(a, b) =_{B[a]} b} \\
  a : \Sigma x : A \cdot B & \vdash a =_{\Sigma x : A \cdot B} \pi_1(a, \pi_2(a))
\end{align*}
\]

One reason for introducing dependent sums is that they facilitate aspects of discourse representation [15, 5]. On the other hand, dependent products support a form of stratified polymorphism. The rigid type structure of standard type theories is often a hindrance to transparent and elegant semantic representation. For example, we cannot in \( G_1 \), uniformly represent the type of the quantifiers as a type of the theory.

\[
\begin{align*}
  \text{Every}_T = \lambda f : T \Rightarrow P \cdot \lambda g : T \Rightarrow P \cdot \forall x : T \cdot f x \Rightarrow g x \\
  \text{Some}_T = \lambda f : T \Rightarrow P \cdot \lambda g : T \Rightarrow P \cdot \exists x : T \cdot f x \wedge g x
\end{align*}
\]

But we can now: the quantifiers \textit{every} and \textit{some} have the following type:

\[ \Pi x : U_0 : (x \Rightarrow P_0) \Rightarrow P_0. \]

7 Our theories and others

We have taken stratified theories beyond the simple set-up introduced in the Ramified framework of Russell and moved matters closer to the rich type structure available in functional programming languages and constructive type theory. Despite their stratified nature, our theories encode a more expressive theory of types than Higher Order Logic (HOL), the standard workhorse of semantics. In particular, they allow a simple generalization to permit \textit{separation/subtypes and dependent types}, and thus pave the way to the inclusion of a form of \textit{stratified polymorphism}.

This brings us to the connection of our intuitionistic theories to CTT. This is rather involved. If, in the above theory, we drop all reference to propositions and add \textit{equality} types, disjoint unions and drop \textit{subtypes}, then we obtain a version of CTT. If to this, we add \textit{a universe of types}, we get a version of CTT with one universe.\(^8\) Furthermore, the theory of last section is similar to a version of CTT with \textit{subtypes/separation types} [19]. But apart from some obvious formal differences in the theory (e.g. no disjoint unions), the way we treat subtypes is distinctive: they have different inferential properties in the grammar and the logic. On the other hand, there are two versions of

\(^8\)The approach to semantics of [5] is based upon this theory but with \textit{Record Types} added. The addition of these do not cause any new technical problems. One of our claims is that the present theory with record types is equally expressive.
the subtype theory in the CTT paradigm: one with strong subtypes and the other with weak ones. We have both, but the weak ones are in the grammar and the strong ones are in the logic. This seems conceptually and technically right. Our intuitionistic theories can be interpreted in CTT using techniques familiar from Feferman’s realizability interpretation of his theories [6].

8 Stratification and liberation

There is also another important conceptual difference. We have given types and propositions back their traditional roles: types now classify objects and propositions take on their traditional role as statement makers. This seems intuitively appropriate for semantics and it brings liberation: we are no longer subject to any form of constructive or antirealist perspective in semantics. Unfortunately, CTT comes as a package; buy one bit and you buy it all. One can argue that one can use CCT in a classical way by defining truth as

\[ T \text{ true iff } \exists x : T \cdot x = x. \]

This will given one classical logic back if the existential quantifier is interpreted classically. But if one does this and unpacks the rules, one is left with something like the theories given here. So unless one is an antirealist about semantics, the main reason for adopting the CTT framework is its expressive power; if one is not, then the theories presented here are liberating. They permit classical devotees to gain the benefit of CTT’s expressive power without having to be antirealist about semantics. On the other hand, the antirealist can use the intuitionistic theories. Everyone is happy.

References

Semantics and Stratification


Received 12 May 2004