CHAPTER 1

Proof Theories and Logical Frameworks

1. Proof-theoretic type theory

1.1. Analytic and synthetic judgement. A synthetic judgement is one for which the experience of *coming to know it* necessarily entails some knowledge which is not implicit in the statement of the judgement; on the other hand, to know an *analytic* judgement is to know it purely on the basis of the information contained inside it. So analytic judgements are decidable, since if they may become evident, it will be purely on the basis of their own content; whereas synthetic judgements become evident to someone when they have obtained some particular evidence for them.

A logical theory has, then, both analytic and synthetic judgements; the judgement $P\ \mathit{prop}$ is analytic, since its evidence follows from the definition of P, whereas the assertion of $P\ \mathit{true}$ entails the knowledge of some extra information, namely a verification of P. When we have extended the logical theory to a type theory in the manner of the previous chapter, the judgement $M \in P$ is also synthetic, since $M \in P$ is not self-evident in general.

But why is it not enough to assert that M verifies P to know whether $M \in P$? It suffices to define a P such that one cannot decide in general whether some term is a verification of it. Let us define the propositional symbol P, and we intend to know the judgement $\mathsf{P}\ \mathit{prop}$, whose meaning is to be expanded as follows:

> To know $\mathsf{P}\ \mathit{prop}$ is to know counts as a canonical verification of P.

We will say, then, that $\bullet$ is a canonical verification of P just when Goldbach's conjecture is true. Then it comes immediately that the judgement $M \in \mathsf{P}$ may not be known or refuted on its own basis, nor even the judgement $\bullet \in \mathsf{P}$, since they depend on a proposition whose truth is not known:

> To know $M \in \mathsf{P}$ is to know that $M \Downarrow M'$ to a canonical verification of P.
>
> $\rightsquigarrow$ To know $M \in \mathsf{P}$ is to know that $M \Downarrow \bullet$ such that Goldbach's conjecture is true.

1.2. Proof of a judgement vs. verification of a proposition. Because the judgement $M \in P$ is synthetic, we cannot say that it gives rise to a proof theory for the logic, since the core judgement of a proof theory $M : A$ must be analytic, in order to avoid the infinite regress of a proof theory requiring a proof theory requiring a proof theory, and so on.

The notion of verification of a proposition could never be the same as proof anyway, except in the most trivial circumstances, since a verification is meant to be an effective operation which realizes the truth of a proposition, and no constraints whatsoever (termination, totality, etc.) are placed on these operations except those which come from the meaning of the judgements (see [**?**], [**?**], [**?**]).

So a proof theory is necessarily intensional, and its judgements are to be analytic/decidable. What is it, then, that we have considered so far which corresponds with a proof M such that $M : P$ in a proof theory? As discussed above, M is not merely a term such that $M \in P$, since this is not in general enough information to know whether M is a proof. In fact, M must comprise all the logical inferences which led to the knowledge that P is true, and so a meaning explanation for the judgement $M : P$ in a proof theory immediately suggests itself:

> To know $M : P$ is to know that M is evidence (demonstration, proof, derivation) of the judgement P *true*.

And so the term domain of the proof theory is not the same as the one that we have considered so far; it must consist in terms which represent traces of the inferences made in the course of knowing the judgements of a logical theory. There is a sense in which one can consider the types of a proof theory to interpret the judgements of the logical theory, and this methodology is called “judgements as types” (and this implies “derivations as terms”).

What I am calling a “proof-theoretic type theory” is a type theory of the sort used in the proof assistants Agda, Coq and Idris, whereas the kind of type theory that I have described in the previous sections, the one based on meaning explanations, underlies the proof assistant Nuprl.

The proof-theoretic type theories on the one hand are often called “intensional” and the computational type theories on the other hand are usually “extensional”; these characterizations are certainly true, though they are not *essential*; moreover, I fear that comparing one of the former with one of the latter is not quite fair, since there is not any clear analogy to be had. That is to say, the judgement $M \in P$ is a judgement which is added to a logical theory and its meaning is (briefly) “M evaluates to a canonical verification of P”, whereas $M : P$ cannot be construed as a judgement added to a logical theory. Instead, it must be understood as part of a (proof) theory which is overlayed atop an existing logical theory; it is possible to understand the theory which contains the judgement $M : P$ to be a metatheory, or logical framework, for the theory which contains the judgement P *true*, which can be construed as the "object language".

In short, the judgements $M \in P$ and $M : P$ are unrelated to each other in two respects: firstly, that they have different meanings, and secondly that the one is at the same level as the judgements of a logical theory, whereas the latter is a judgement in a theory which is defined over a logical theory.

2. Martin-Löf's equational logical framework

To make this more concrete, let us expound a proof theoretic type theory called **MLLF**, which stands for "Martin-Löf's (equational) logical framework";[1] in the course of introducing each type, we will specify which judgement of the underlying logical theory it is meant to interpret.

We start with four categorical judgements:

Judgement Form	Pronunciation
$\alpha : \mathit{type}$	α is a type
$\alpha = \beta : \mathit{type}$	α and β are equal types
$M : \alpha$	M is of type α
$M = N : \alpha$	M and N are equal at type α

But we have not defined the meaning of the judgements; let us do so below:

> To know $\alpha : \mathit{type}$ is to know what counts as an object of type α, and when two such objects are equal.

For now, we'll leave the question of what is an "object" as abstract; in many cases, types will represent judgements of a logical theory, and the objects will be the derivations (demonstrations, proofs) of those judgements.

> To know $\alpha = \beta : \mathit{type}$ is to know that any object of type α is also an object of type β, and two equal objects of type α are equal as objects of type β (necessarily presupposing $\alpha : \mathit{type}$ and $\beta : \mathit{type}$).
>
> To know $M : \alpha$ is to know that M is an object of type α (necessarily presupposing $\alpha : \mathit{type}$).
>
> To know $M = N : \alpha$ is to know that M and N are equal objects of type α (necessarily presupposing $M : \alpha$ and $N : \alpha$).

In addition to the above judgements, we will need contexts (with their well-formedness judgement Γ *ctx*) and an intensional sequent judgement $\boxed{\Gamma \vdash \mathcal{J}}$; their

[1] For a detailed overview of Martin-Löf's equational logical framework, see [**?**].

meanings here will differ from the sequent judgements of computational type theory, in that they must mean proof-theoretic derivability, rather than semantic consequence.

At this point, we may begin adding types to the logical framework. In practice, most types which we will introduce in the logical framework will be defined in terms of a judgement of the logical theory which lies below it. For instance, hypothetical judgement in the logical theory is represented by a function type in the logical framework, $(x : \alpha)\beta$, whose typehood is meant to be evident under the following circumstances

$$\frac{\alpha : \mathit{type} \quad x : \alpha \vdash \beta : \mathit{type}}{(x : \alpha)\beta : \mathit{type}}$$

Or as a hypothetical judgement, $(x : \alpha)\beta : \mathit{type}\ (\alpha : \mathit{type}, x : \alpha \vdash \beta : \mathit{type})$.

Now, to know this judgement is to know that under the circumstances we know what is an object of type α and when two such objects are equal, and that if we have such an object x of type α, we know what an object of type β is, and when two such objects are equal—then we know what an object of type $(x : \alpha)\beta$ is, and moreover, for any two objects y, z of type α, that $[y/x]\beta$ and $[z/x]\beta$ are equal as types. To make this evident, then, we will say that under those circumstances an object of type $(x : \alpha)\beta$ is an object $[x]M$ such that one knows $x : \alpha \vdash M : \beta$ and $|_{x,y}\ [y/x]M = [z/x]M : [y/x]\beta\ (y = z : \alpha)$; furthermore, two such objects are equal just when they yield equal outputs for equal inputs.

Then, for each atomic proposition P, we can easily define a type $\mathsf{Prf}(P)$, as follows. Under the circumstances that P *prop* in the logical theory, then $\mathsf{Prf}(P)$: *type* in the logical framework, since we will define an object of type $\mathsf{Prf}(P)$ to be a derivation of P *true*; beyond reflexivity, further definitional equalities can be added to reflect the harmony of introduction and elimination rules.

Now, the definitions we have given for the types above are "intuitively" correct, but they actually fail to satisfy the meaning explanation that we have given for α : *type*, because they do not take into account neutral terms. In the following sections, we will investigate this problem in more detail and propose a solution.

2.1. What is an "object"? It is time to revisit what it means to be an "object" of a type in the proof-theoretic type theory; we must note how this will necessarily differ from what it meant to be a "verification" of a proposition in the previous sections. Namely, a verification of a proposition is either a *canonical verification* of that proposition (and what sort of thing this might be is known from the presupposition P *prop*), or it is a means of getting such a canonical verification (i.e. a term which evaluates to a canonical verification).

On the other hand, what we have called an "object" of type P is quite different, since in addition to the possibility that it is a canonical proof of the judgement

P *true*, it may also be *neutral* (i.e. blocked by a variable); we will call this "normal" rather than "canonical". Why does this happen?

In order to keep the judgement $M : A$ analytic (decidable), its meaning explanation can no longer be based on the idea of the computation of closed terms to canonical form; instead, we will consider the computation of open terms (i.e. terms with free variables) to *normal* form. The desire for $M : A$ to be analytic follows from our intention that it characterize a *proof theory*: we must be able to recognize a proof when we see one. But why are closed-term-based meaning explanations incompatible with this goal? Consider briefly the following judgement:

$$|_n\, M(n) \in P \ (n \in \mathbb{N})$$

To know this judgement is to know that $M(n)$ computes to a canonical verification of P whenever n is a natural number; when P's use of n is not trivial, this amounts to testing an infinite domain (all of the natural numbers), probably by means of mathematical induction. The judgement is then clearly synthetic: to know it is, briefly, to have come up with an (inductive) argument that $M(N)$ computes to a canonical verification of P at each natural number n.

On the other hand, the judgement $n : \mathbb{N} \vdash M(n) : P$ must have a different meaning, one which admits its evidence or refutation purely on syntactic/analytic grounds. In essence, it is to know that $M(n)$ is a proof of P for any *arbitrary* object/expression n such that $n : \mathbb{N}$ (i.e., the only thing we know about n is that it is of type $\mathbb{N}$; we do not necessarily know that it is a numeral).

3. A critique of MLLF

The type theory which we constructed in the previous section is to be considered a proof theory for a logic with the judgements P *prop*, P *true* and $\mathcal{J}$ $(\mathcal{J}')$. There are a few reasons to be dissatisfied with this state of affairs, which I shall enumerate in this section.

3.1. Lack of computational content. Unlike the type theory in the first chapter, there is no built-in computational content. In a computational type theory which is defined by the verificationist meaning explanations, the computational content of terms is understood immediately by means of the $M \Downarrow M'$ relation; that is, computation is prior to the main judgements because their meaning explanations are defined in terms of evaluation to canonical form.

On the other hand, in the type theory above we did not give a primitive reduction relation; instead, we simply permitted the endowement of proofs with definitional equalities which reflect the harmony of introduction and elimination rules. That is, if we have known the judgement P *true* by means of an indirect argument (derivation), it must be the case that this derivation corresponds to a direct one; we reflect this in the proof theory by defining the indirect derivation to be definitionally equal to the direct one.

However, this does not amount to computational content being present in terms: only *post facto* may the definitional equality be construed as giving rise to computation, through a metamathematical argument which shows that the definitional equality is confluent and can be used to define a functional normalization relation.

And this is the reason for the peculiarity of the proof-theoretic meaning explanations, namely that they do not include phrases like "evaluates to a canonical...", since evaluation may only be understood after taking the meanings of the judgements ($\alpha : \mathit{type}$, $\alpha = \beta : \mathit{type}$, $M : \alpha$, $M = N : \alpha$) as giving rise to a closed formal system which is susceptible to metamathematical argument: to refer to evaluation in the meaning explanations for the core judgements, then, would be impredicative.

3.2. Modularity of definition. By the same token, the distinction between canonical (direct) and non-canonical (indirect) proof may not be understood as a core notion in the theory, but must be understood separately, secondarily. Why is this a problem? It means that the definition of each type must be made with the full knowledge of the definitions of every other type; in essence, the open-ended nature of type theory is obliterated and one is forced into a fixed formal system; this is in addition to the fact that it causes the epistemic content of $\alpha : \mathit{type}$ for any type α to be extremely complicated.

To illustrate, let us consider as an example a type theory which has four type-formers: trivial truth $\top$, trivial falsity $\bot$, implication $(\alpha)\beta$, and conjunction $\alpha\&\beta$; we will then introduce the following terms to represent proofs: the trivial element $\bullet$, *reductio ad absurdum* $\mathsf{abort}(\alpha; E)$, abstraction $[x : \alpha]E$, application $E(E')$, pairing $\langle E, E'\rangle$, and projections $\mathsf{fst}(E)$, $\mathsf{snd}(E)$.

If we will try to make the judgement $\top : \mathit{type}$ evident, the deficiencies of the formulation will immediately present themselves.

> To know $\top : \mathit{type}$ is to know what counts as an object of type $\top$, and when two such objects are equal. An object of type $\top$, then, is either the expression $\bullet$, or an expression $\mathsf{abort}(\top; E)$ such that we know $E : \bot$, or an expression $E(E')$ such that we know $E : (\alpha)\top$ and $E' : \alpha$, or an expression $\mathsf{fst}(E)$ such that we know $E : \top\&\beta$ for some β, or an expression $\mathsf{snd}(E)$ such that we know $E : \alpha\&\top$ for some α; and we additionally have that $\bullet$ is equal to $\bullet$, and ...

To save space, we elide the rest of the definition of equality for $\top$; what we have seen so far already suffices to bring to light a serious problem: the definition of any type requires knowledge of the entire syntax of the theory. The judgement $\alpha : \mathit{type}$

may never be made evident in isolation, but must be done with full understanding of all the other types and their definitions.

Furthermore, to extend an existing theory with a new type, the definitions of every other type must be rewritten to account for the elimination forms of the new type.

4. A way forward: verifications & uses

The second critique of **MLLF** may be partially addressed by fragmenting type theory into a logic of *verifications & uses*: instead of a type being defined by its introduction rules, it must be simultaneously defined by its introduction rules (verifications) and its elimination rules (uses). In practice, this amounts to a standard technique known as *bidirectional type checking*.

The semantic priority of the forms of judgement also changes drastically: the sequent judgement must in this case be explained *before* the categorical judgements; moreover, sequents may no longer be explained modularly in terms of general and hypothetical judgement, since the latter amounts to *semantic* consequence (admissibility), whereas the meaning of a sequent in a proof theory should be *syntactic* consequence (derivability).

Because the target theory lacks computation, it is necessary to rule out redexes from terms syntactically, but this complicates the definition of substitution; to address this, Watkins introduced in [**?**] a technique known as *hereditary substitution*, which is a family of syntax-directed (algorithmic) judgements which contract redexes along the way, guaranteeing canonical form in their outputs. Both bidirectional type checking and hereditary substitutions have been used to great effect in the descendants of the Edinburgh Logical Framework (see [**?**]).

The first critique, the lack of computational content, is more difficult to address. Roughly, the right way to do it is to replace the notion of the evaluation of closed terms to canonical form with the evaluation of *open* terms to normal form. Peter Dybjer demonstrates in [**?**] how this technique may be used to endow the Calculus of Constructions with a meaning explanation, albeit necessarily of a very different kind than we have considered here.