# TYPE THEORY AND ITS MEANING EXPLANATIONS

JONATHAN STERLING

Abstract. At the heart of intuitionistic type theory lies an intuitive semantics called the "meaning explanations"; crucially, when meaning explanations are taken as definitive for type theory, the core notion is no longer "proof" but "verification". Well explore how type theories of this sort arise naturally as enrichments of logical theories with further judgements, and contrast this with modern proof-theoretic type theories which interpret the judgements and proofs of logics, not their propositions and verifications.

To start, we will consider the notion of a *logical theory*; in my mind, it starts with a species (or set) of judgements that can be proposed, asserted, and (if they are evident) known.

## 1. Judgements of a Logical Theory

The basic forms of judgement for a logical theory will be $\boxed{P\ \mathit{prop}}$ and $\boxed{P\ \mathit{true}}$; and what is $P$? It is a member of the species of terms, which are made meaningful in the course of making the judgement $P\ \mathit{prop}$ evident for a proposition $P$.

To each judgement is assigned a *meaning explanation*, which explicates the knowledge-theoretic content of the judgement. For a judgement $\mathcal{J}$, a meaning explanation should be in the form:

> To know $\mathcal{J}$ is to know...

The meaning of the judgement $P\ \mathit{prop}$ is, then, as follows:

> To know $P\ \mathit{prop}$ is to know that $P$ is a proposition, which is to know what counts as a direct verification of $P$.

So if a symbol $P$ is taken to denote a proposition, we must know *what sort of thing* is to be taken as a direct verification of $P$, and this is by definition. A "direct verification" is understood in constrast with an "indirect verification", which is to be thought of as a means or plan for verifying the proposition. Now, the judgement $P\ \mathit{true}$ is only meaningful in case we know $P\ \mathit{prop}$ (this is called a presupposition). Then the meaning of $P\ \mathit{true}$ is as follows:

> To know $P\ \mathit{true}$ is to have a verification of $P$.

From the (implicit) presupposition $P\ \mathit{prop}$, we already know what counts as a verification, so the meaning explanation is well-defined. Note that it is the same to

Thanks to Bob Harper, Peter Dybjer and Bengt Norström for invaluable conversations that helped to clarify my understanding of the meaning explanations for Martin-Löf's Type Theory.

have a means or plan for verifying $P$ as to have a (direct) verification; this follows from the fact that one may put into action a plan for verifying $P$ and get out such a verification, and likewise, it is possible to propound a plan of verification by appealing to an existing verification.

The judgements we have described so far are "categorical" in the sense that they are made without assumption. We will need to define a further form of judgement, which is called "hypothetical", and this is the judgement under hypothesis $\boxed{\mathcal{J}\ (\mathcal{J}_1, \ldots, \mathcal{J}_n)}$. And its meaning explanation is as follows:

> To know the judgement $\mathcal{J}\ (\mathcal{J}_1, \ldots, \mathcal{J}_n)$ is to know the categorical judgement $\mathcal{J}$ when you know the judgements $\mathcal{J}_1, \ldots, \mathcal{J}_n$.

## 2. Propositions

Now that we have propounded and explained the minimal system of judgements for a logical theory, let us populate it with propositions. First, we have falsity $\bot$, and we wish to make $\bot$ *prop* evident; to do this, we simply state what counts as a direct verification of $\bot$, which is that there can be no direct verification of $\bot$.

The next basic proposition is trivially true $\top$, and to make $\top$ *prop* evident, we state that a direct verification of $\top$ is trivial. The meaning of $\top$ thus validates the judgement $\top$ *true*.

Next, let us define conjunction; in doing so, we will make evident the hypothetical judgement $P \wedge Q$ *prop* ($P$ *prop*, $Q$ *prop*); equivalently, we can display this as a rule of inference:

$$\frac{P\ \mathit{prop} \quad Q\ \mathit{prop}}{P \wedge Q\ \mathit{prop}}$$

A direct verification of $P \wedge Q$ consists in a verification of $P$ and a verification of $Q$; this validates the assertion of the judgement $P \wedge Q$ *true* ($P$ *true*, $Q$ *true*). Because it is a valid inference, we can write it as an inference rule:

$$\frac{P\ \mathit{true} \quad Q\ \mathit{true}}{P \wedge Q\ \mathit{true}}$$

A direct verification of $P \vee Q$ may be got either from a verification of $P$ or one of $Q$. From this definition we know $P \vee Q$ *prop* ($P$ *prop*, $Q$ *prop*), or

$$\frac{P\ \mathit{prop} \quad Q\ \mathit{prop}}{P \vee Q\ \mathit{prop}}$$

The verification conditions of disjunction give rise to two evident judgements $P \vee Q$ *true* ($P$ *true*) and $P \vee Q$ *true* ($Q$ *true*), which we can write as inference rules:

$$\frac{P\ \mathit{true}}{P \vee Q\ \mathit{true}} \qquad \frac{Q\ \mathit{true}}{P \vee Q\ \mathit{true}}$$

Finally, we must define the circumstances under which $P \supset Q$ is a proposition (i.e. when $P \supset Q$ *prop* is evident). And we intend this to be under the circumstances that $P$ is a proposition, and also that $Q$ is a proposition assuming that $P$ is true. In other words, $P \supset Q$ *prop* $(P$ *prop*$, Q$ *prop* $(P$ *true*$))$, or

$$\frac{P \text{ prop} \quad Q \text{ prop } (P \text{ true})}{P \supset Q \text{ prop}}$$

Now, to validate this judgement will be a bit more complicated than the previous ones. But by unfolding the meaning explanations for hypothetical judgement, proposition-hood and truth of a proposition, we arrive at the following explanation:

> To know $P \supset Q$ *prop* $(P$ *prop*$, Q$ *prop* $(P$ *true*$))$ is to know what counts as a direct verification of $P \supset Q$ when one knows what counts as a direct verification of $P$, and, when one has such a verification, what counts as a direct verification of $Q$.

(Note that unless $P$ *true*, it need not be evident that $Q$ *prop*.) Now, if this judgement is going to be made evident, then we must indeed come up with what should count as a direct verification of $P \supset Q$ under the assumptions described above.

And so to have a direct verification of $P \supset Q$ is to have a verification of $Q$ assuming that one has one of $P$; this is the meaning of implication, and it validates the judgement $P \supset Q$ *true* $(Q$ *true* $(P$ *true*$))$ (elliding the hypothesis for propositionhood), and may be written as an inference rule as follows:

$$\frac{Q \text{ true } (P \text{ true})}{P \supset Q \text{ true}}$$

## 3. Judgements for Verifications

So far, we have given judgements which circumscribe what it means to be a proposition, and thence for each proposition, we have by definition a notion of what should count as a verification of that proposition. And by definition, to assert the judgement $P$ *true* is to assert that one has a verification of $P$, but we have not considered any judgements which actually describe such verifications.

It is a hallmark of Martin-Löf's program to resolve the contradiction between syntax and semantics not by choosing symbols over meanings or meanings over symbols, but by endowing symbols with meaning in the course of knowing the evident judgements. As such $P$ is a symbol, but when we assert $P$ *prop* we are saying that we know what proposition $P$ denotes.

A similar thing can be done with verifications themselves, by placing them in the syntactic domain together with the propositional symbols. And then, we can consider a judgement such as "$M$ is a verification of $P$", and in knowing that

judgement, we know what verification $M$ is meant to denote. In practice, this judgement has been written in several ways:

| | |
|---|---|
| $M \in P$ | $M$ is an element of $P$ |
| $M \Vdash P$ | $M$ realizes $P$ |
| $P \llcorner \mathsf{ext}\ M \lrcorner$ | $P$ is witnessed by $M$ |

But they all mean the same thing, and so we will tentatively give the following meaning explanation to this new judgement:

> * To know $M \in P$ is to know that $M$ is a verification of $P$.

But now that we have started to assign expressions to verifications, we must be more careful about differentiating *direct verifications* (which we will call "canonical") from *indirect verifications* (which we will call "non-canonical"). So the domain of expressions must itself be accorded with a notion of reduction to canonical form, and this corresponds with putting into action a plan of verification in order to get an actual verification.

> To know $M \Rightarrow M'$ is to know that $M$ is an expression which reduces to a canonical form $M'$.

Now, we can rewrite the previous meaning explanation as follows:

> To know $M \in P$ is to know an $M'$ such that $M \Rightarrow M'$ and $M'$ is a canonical (direct) verification of $P$.

The meaning explanation for $P\ \mathit{prop}$ must be accordingly modified to take into account the computational behavior of the expression domain:

> To know $P\ \mathit{prop}$ is to know a $P'$ such that $P \Rightarrow P'$ and $P'$ is a canonical proposition, which is to say, that one knows what counts as a canonical verification for $P'$.

As an example, then, we will update the evidence of the assertion $P \supset Q\ \mathit{prop}\ (P\ \mathit{prop}, Q\ \mathit{prop}\ (P\ \mathit{prop}))$. The meaning of this, expanded into spoken language, is as follows:

> To know $P \supset Q\ \mathit{prop}\ (P\ \mathit{prop}, Q\ \mathit{prop}\ (P\ \mathit{prop}))$ is to know what counts as a canonical verification of $P \subset Q$ under the circumstances that $P \Rightarrow P'$, such one knows what counts as a canonical verification $P'$, and, if one has such a verification, $Q \Rightarrow Q'$ such that one knows what counts as a canonical verification of $Q'$.

And this is evident, since we will say that a canonical verification of $P \subset Q$ is an expression $\lambda x.E$ such that we know $E \in Q\ (x \in P)$. This validates the assertion $\lambda x.E \in P \supset Q\ (E \in Q\ (x \in P))$, written as an inference rule:

$$\frac{E \in Q\ (x \in P)}{\lambda x.E \in P \supset Q}$$

By the addition of this judgement, we have graduated from a logical theory to a type theory, in the sense of *Constructive Mathematics and Computer Programming* (Martin-Löf, 1979). In fact, we may dispense with the original $P$ *true* form of judgement by *defining* it in terms of the new $M \in P$ judgement as follows:

$$\frac{M \in P}{P \ true}$$

Further forms of judgement, such as assertions of equality between propositions and verifications, may be added, as Martin-Löf does. However, this is not strictly necessary as they too may be defined in terms of the $M \in P$ judgement in the presence of an *equality* proposition; this is what is done in Constable et al's Computational Type Theory, which has only one primitive form of judgement.

[TODO] Give the syntax of full type theory and expand the meaning explanations to account for new propositions and verifications

## 4. Proof-Theoretic Type Theory

4.1. **Analytic and Synthetic Judgement.** A synthetic judgement is one for which the experience of *coming to know it* necessarily results in the synthesis of some new information; 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 when one has acquired some particular evidence for them.

A logical theory has, then, both analytic and synthetic judgements; the judgement $P$ *prop* is analytic, since its evidence follows from the definition of $P$, whereas the assertion of $P$ *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 sections, 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 $M \in P$? 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 $\mathsf{P}$, and we intend to know the judgement $\mathsf{P}$ *prop*, whose meaning is to be expanded as follows:

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

We will say, then, that $\bullet$ is a canonical verification of $\mathsf{P}$ 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 an proposition whose truth is not known:

> To know $M \in \mathsf{P}$ is to know that $M \Rightarrow M'$ to a canonical verification of $P$.

$\Leftrightarrow$ To know $M \in \mathsf{P}$ is to know that $M \Rightarrow \bullet$ such that Goldbach's conjecture is true.

4.2. **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 the proposition, and no constraints whatsoever (termination, totality, etc.) are placed on these operations except those which come from the meaning of the judgements (see Dummett).

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 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 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 meaning-theoretic type theories on the other hand are usually "extensional"; these characterizations are certainly true, though 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 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 the latter is a judgement in a theory which is defined over a logical theory.

4.3. **Meaning explanations for proof theoretic type theory.** To make this more concrete, let us expound a proof theoretic type theory called **MLLF**, which stands for "Martin-Löf's Logical Framework"; in the course of introducing each type, we will specify which judgement of the underlying logical theory $\mathcal{L}$ it is meant to interpret. Since we are now beginning to refer to judgements in multiple theories, where there is ambiguity, we will use the notation "$\mathcal{T}$-judgement" to refer to a judgement in the theory $\mathcal{T}$, and $\mathcal{T} \vdash \mathcal{J}$ will be shorthand for "the $\mathcal{T}$-judgement $\mathcal{J}$ is evident".

We start with four categorical judgements:

| Judgement | Pronunciation |
|---|---|
| $\alpha$ : *type* | $\alpha$ is a type |
| $\alpha = \beta$ : *type* | $\alpha$ and $\beta$ are equal types |
| $M : \alpha$ | $M$ is of type $\alpha$ |
| $M = N : \alpha$ | $M$ and $N$ are equal at type $\alpha$ |

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

> To know $\alpha$ : *type* is to know what counts as an object of type $\alpha$, 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 $\mathcal{L}$-judgements and objects will represent $\mathcal{L}$-derivations.

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

Next, we will introduce hypothetical judgement in the logical framework; to avoid confusion, where $\mathcal{J}$ ($\mathcal{J}'$) is a hypothetical $\mathcal{L}$-judgement, we will use $\mathcal{J}$ $[\mathcal{J}']$ for hypothetical judgements in the logical framework. Then,

$$\text{To know } \mathcal{J}\ [\mathcal{J}_1, \ldots, \mathcal{J}_n] \text{ is to know } \mathcal{J} \text{ when one knows } \mathcal{J}, \ldots, \mathcal{J}_n.$$

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 $\mathcal{L}$ which lies below it. For instance, hypothetical judgement in $\mathcal{L}$ is represented by a function type in the logical framework, $(x : \alpha)\beta$, whose typehood is meant to be evident under the following circumstances

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

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

Now, to know this judgement is to know that under the circumstances we know what is an object of type $\alpha$ and when two such objects are equal, and that if we have such an object $x$, we know what an object of type $\beta$ is, and when two such objects are equal—then we know what an object of type $(x : \alpha)\beta$ is. 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 $M : \beta\ [x : \alpha]$ and $[y/x]M = [z/x]M\ [y = z : \alpha]$; furthermore, two such objects are equal just when they yield equal outputs for equal inputs. (The equality is extensional, but since it is with respect to the definitional equality, this does not break the decidability of the judgement)

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

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 the presupposition $P\ \mathit{prop}$), 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 $P\ \mathit{true}$, 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:

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

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 $M(n) : P \ [n : \mathbb{N}]$ 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).

4.3.1. *Finer Judgements for Proof-Theoretic Type Theory.* As soon as we have begun to consider the computation of open terms to normal form, as opposed to the computation of closed terms to canonical form, a more fine-grained structure of judgements begins to present itself. We'll replace the $M : \alpha$ judgement with two new ones:

| | |
|---|---|
| $R \uparrow \alpha$ | $R$ **synthesizes** type $\alpha$ |
| $M \downarrow \alpha$ | $M$ **checks** type $\alpha$ |

A peculiarity of the notation is that $R \uparrow \alpha$ is really a synthetic judgement involving the expression $R$ only, where the evidence for the judgement contains $\alpha$. Its meaning explanation is, "I know of what type $R$ is (and by the way, it is $\alpha$)". On the other hand, $M \downarrow \alpha$ is analytic, and it carries the epistemic content of the old $M : \alpha$ judgement; to know it is to know that $M$ computes to a normal proof of $\alpha$.

Let us amend the meaning explanation for the judgements of type theory; first, typehood, which will once again be cast in terms of canonical forms (of proofs, not verifications):

> To know $\alpha : \mathit{type}$ is to know what counts as a canonical proof of $\alpha$, and when two proofs of $\alpha : \mathit{type}$ are equal.

Then the typing judgements are given:

> To know $R \uparrow \alpha$ is to know that $R \Rightarrow R'$ such that $R'$ is a non-canonical proof of a known type, namely $\alpha$.

> To know $M \downarrow \alpha$ is to know that $M \Rightarrow M'$ such that $M'$ is a normal proof of $\alpha$ (i.e., it is either a canonical proof of $\alpha$ or it is a neutral term such that one knows $M \uparrow \alpha$).

We will restrict the hypothetical judgement to only contain hypotheses of the form $x \uparrow A$ where $x$ is a variable, or $\alpha : \mathit{type}$ where $\alpha$ is a variable, or further hypothetical judgements:

> To know $\mathcal{J}\ [\mathcal{J}_1, \ldots, \mathcal{J}_n]$ is to know $\mathcal{J}$ when one knows $\mathcal{J}_1, \ldots, \mathcal{J}_n$.

Now, we will show how to make true the judgement $\alpha : \mathit{type}$ for a number of types $\alpha$, by defining them. For example, to define conjunction, first we introduce the following syntax: $\alpha \& \beta$, $\langle M, N \rangle$, $\mathsf{fst}(R)$, $\mathsf{snd}(R)$ subject to the following computation rules:

$$\mathsf{fst}(\langle M, N \rangle) \Rightarrow M \qquad \mathsf{snd}(\langle M, N \rangle) \Rightarrow N$$

$$\frac{R \Rightarrow R' \quad R' \text{ is neutral}}{\mathsf{fst}(R) \Rightarrow \mathsf{fst}(R')} \qquad \frac{R \Rightarrow R' \quad R' \text{ is neutral}}{\mathsf{snd}(R) \Rightarrow \mathsf{snd}(R')}$$

The, to make the judgement $\alpha \& \beta : \mathit{type}\ [\alpha : \mathit{type}, \beta : \mathit{type}]$ evident, let's first expand its meaning explanation.

> To know $\alpha \& \beta : \mathit{type}\ [\alpha : \mathit{type}, \beta : \mathit{type}]$ is to know what counts as a canonical proof of $\alpha \& \beta$, and when two proofs of $\alpha \& \beta$ are equal, when one knows that what counts as a canonical proof of $\alpha$ and when two proofs of $\alpha$ are equal, and one knows what counts as a canonical proof of $\beta$ and when two proofs of $\beta$ are equal.

This is evident, since we will say that a canonical proof of $\alpha \& \beta$ is a pair $\langle M, N \rangle$ such that we know $M \downarrow \alpha$ and $N \downarrow \beta$, and that the equality of pairs is structural. This corresponds to the following inference rules:

$$\frac{M \downarrow \alpha \quad N \downarrow \beta}{\langle M, N \rangle \downarrow \alpha \& \beta} \qquad \frac{M = M' : \alpha \quad N = N' : \beta}{\langle M, N \rangle = \langle M', N' \rangle : \alpha \& \beta}$$

Based on these definitions and computation rules, the following inference rules are justified:

$$\frac{R \uparrow \alpha\&\beta}{\mathsf{fst}(R) \uparrow \alpha} \qquad \frac{R \uparrow \alpha\&\beta}{\mathsf{snd}(R) \uparrow \beta}$$

$$\frac{M \downarrow \alpha\&\beta}{M = \langle \mathsf{fst}(M), \mathsf{snd}(M) \rangle : \alpha\&\beta}$$