Propositions as [types]  
(extended abstract)

Steve Awodey* and Andrej Bauer**

Abstract. We consider a modal operator \([A]\) on types for erasing computational content and formalizing a notion of proof irrelevance. We give rules for such “bracket types” in dependent type theory and provide complete semantics using regular categories and topological models. We also show how to interpret first-order logic in type theory with brackets, and we make use of the translation to compare type theory with intuitionistic and with classical first-order logic.

Keywords: modal logic, category theory, type theory, regular categories, intuitionistic logic, bracket types

1 Introduction

According to the Curry-Howard “Propositions as types” conception of the theory of types, propositions and types are identified:

\[
\text{Propositions} = \text{Types}.
\]

In this work we distinguish propositions and types, while staying within a type-theoretic framework. Specifically, we regard only some types as propositions. Additionally, each type \(A\) has an associated proposition \([A]\), giving a correspondence:

\[
\frac{\text{Propositions}}{\text{Types}} \xrightarrow{[-]} \frac{\text{[Types]}}{}
\]

which is in fact an adjunction. Since it will turn out that \([P] = P\) for any proposition \(P\), the propositions are exactly the types in the image of the bracket constructor \([-]\), the bracket types. Thus we have:

\[
\text{Propositions} = [\text{Types}].
\]

* Department of Philosophy, Carnegie Mellon University, Pittsburgh, USA. E-mail: awodey@cmu.edu
** Institute of Mathematics, Physics and Mechanics, University in Ljubljana, Slovenia. E-mail: Andrej.Bauer@andrej.com
We can think of a bracket type \([A]\) as the proposition whose proofs are the terms of type \(A\). Another way to understand \([A]\) is to think of it as the “extension of \(A\)”, whereas a computational interpretation of \([A]\) would be “the type \(A\) with the computational content erased”.

The bracket types which we consider are essentially the same as the \textit{mono types} of Maietti [Mai98], in a suitable setting, Palmgren [Pal01] formulated a BHK interpretation of intuitionistic logic and used \textit{image factorizations}, which are used in the semantics of our bracket types, to relate the BHK interpretation to the standard category-theoretic interpretation of propositions as subobjects. Aczel and Gambino [AG01] have promoted what they call \textit{logic-enriched type theory} in which they separate the logic from type theory. The bracket types can be used to translate the primitive logic back into the type theory (the usual translation “propositions as types” works as well). Already in his \textit{Dialectica} article, Lawvere [Law69] proposed a categorical treatment of proof theory that is closely related to bracket types. The work of Mendler [Men90] is also related to this topic.

The bracket is a diamond operation, in the sense of modal logic, in a system with \textit{dependent} types. As such, it is an example of quantified modal logic. One can consider extending the work of Moggi [Mog91] and others on modal type theory to the dependent case. See also [DP00] in this connection.

2 Bracket Types

We consider a Martin-Löf style dependent type theory [ML84,ML98] with \textit{strong} and \textit{extensional} equality and \textit{strong} dependent sums, cf. [Jac99]. For reference, we list the rules in Appendix A.

Among the types, there are some that satisfy the following condition of “\textit{proof irrelevance}”:

\[
\frac{\Gamma \vdash P \text{ type } \quad \Gamma \vdash q : P \quad \Gamma \vdash p : P}{\Gamma \vdash p = q : P}
\]

We call these types \textit{propositions}, and we introduce a new type constructor \([-\]\) associating to each type \(A\) a proposition \([A]\). The axioms given in Figure 1 are designed with the following adjunction in mind, for any type \(A\) and proposition \(P\):

\[
\frac{x : A \vdash p : P}{x' : [A] \vdash p' : P}
\]
This states that the bracket operation is left adjoint to the inclusion of propositions into types. In the notation introduced in Figure 1, we can take \( p' = (p \text{ where } [x] = x') \), since the equality condition on \( p : P \) for elimination is satisfied by proof irrelevance (1).

As an example, let us show that the term forming operation \([-\cdot]\) is 'epic' in the following sense:

\[
\frac{\Gamma, x:A \vdash s([x]/u) = t([x]/u) : B}{\Gamma, u:A \vdash s = t : B}
\]  

(3)

If we think of a term \( \Gamma, x:A \vdash r : B \) as an arrow \( A \to B \) in the slice category over \( \Gamma \), as we will in Section 3, then we have the following situation over \( \Gamma \):

\[
A \xrightarrow{[-\cdot]} [A] \xrightarrow{s} B
\]

Now (3) says that \( s \circ [-\cdot] = t \circ [-\cdot] \) implies \( s = t \) for arbitrary \( s, t : A \to B \), which means that \([-\cdot] \) is epic. To prove (3), observe first that by the equality rule we have

\[
\Gamma, x:A, y:A \vdash [x] = [y] : [A]
\]

therefore

\[
\Gamma, x:A, y:A \vdash s([x]/u) = s([y]/u) : [A]
\]

which means that we can form the term \( s([x]/u) \text{ where } [x] = u \). Similarly, we can form the term \( t([x]/u) \text{ where } [x] = u \). Now we get

\[
s =_\eta (s([x]/u) \text{ where } [x] = u) = (t([x]/u) \text{ where } [x] = u) =_\eta t.
\]

The second equality follows from the assumption \( s([x]/u) = t([x]/u) \) and the compatibility rule for \textit{where} terms.

A consequence of (3) is the following conversion, called \textit{exchange}:

\[
b \text{ where } [x] = (p \text{ where } [y] = q) = (b \text{ where } [x] = p) \text{ where } [y] = q.
\]

The rule is valid when \( y \neq x \) and \( y \not\in \text{FV}(b) \). By (3) it suffices to verify the exchange rule for the case \( q = [z] \) where \( z:A \) is a fresh variable.

The elimination rule involving the \textit{where} term is perhaps unsatisfactory from the type-theoretic point of view, since it involves an equality judgement as a premise. One could consider a different formulation of
Bracket types

\[
\begin{align*}
\Gamma \vdash A & \quad \text{type formation} \quad \Gamma \vdash a : A & \quad \text{intro} \\
\Gamma \vdash [A] & \quad \text{type} \quad \Gamma ; x : A ; b : B & \quad \Gamma ; x : A ; y : A \vdash b[y/x] : B & \quad \text{elim} \\
\Gamma \vdash q : [A] & \quad \Gamma \vdash B & \quad \Gamma ; x : A \vdash b & \quad \Gamma \vdash b \text{ where } [x] = q : B & \\
\Gamma \vdash p : [A] & \quad \Gamma \vdash q : [A] & \quad \text{equality} \\
\Gamma \vdash p = q : [A] &
\end{align*}
\]

Conversions

\[
\begin{align*}
b \text{ where } [x] = [a] & \quad =_{\beta} \quad b[a/x] \\
b[x/u] & \quad =_{\eta} \quad b[q/u]
\end{align*}
\]

Free variables

\[
\begin{align*}
\text{FV}(\{A\}) & \quad = \text{FV}(A) \\
\text{FV}(\{a\}) & \quad = \text{FV}(a) \\
\text{FV}(b \text{ where } [x] = q) & \quad = (\text{FV}(b) \setminus \{x\}) \cup \text{FV}(q)
\end{align*}
\]

Substitution

\[
\begin{align*}
[A] t/x] & \quad = [A[t/x]] \\
[a] t/x] & \quad = [a(t/x)] \\
b \text{ where } [x] = q & \quad t/y] \quad = b[t/y] \text{ where } [x] = (q(t/y)) & \\
\text{(provided } x \neq y \text{ and capture of } x \text{ in } t \text{ is avoided})
\end{align*}
\]

Compatibility rules

\[
\begin{align*}
A = A' & \quad \implies \quad [A] = [A'] \\
a = a' & \quad \implies \quad [a] = [a'] \\
b = b' \land q = q' & \quad \implies \quad (b \text{ where } [x] = q) = (b' \text{ where } [x] = q')
\end{align*}
\]

\textbf{Fig. 1.} Bracket types
bracket types in which there is a new judgment “\( \Gamma \vdash P \text{ prop} \)”, expressing the fact that \( P \) is a proposition. The rules would then be as follows:

\[
\begin{align*}
\Gamma \vdash A \text{ type} & \quad \Gamma \vdash P \text{ prop} \\
\frac{\Gamma \vdash [A] \text{ prop}}{\Gamma \vdash P \text{ type}}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash a : A & \quad \Gamma \vdash [A] \text{ prop} \\
\frac{\Gamma \vdash q : [A]}{\Gamma \vdash p \text{ where } [x] = q : P}
\end{align*}
\]

As stated so far, however, the rules permit many different interpretations, including the trivial one in which the only proposition is the unit type \( 1 \), and \([A] = 1\) for all \( A \). One could additionally assert that certain types are propositions, e.g.:

\[
\begin{align*}
\Gamma \vdash 0 \text{ prop} & \quad \Gamma \vdash 1 \text{ prop} \\
\frac{\Gamma \vdash P \text{ prop} \quad \Gamma \vdash Q \text{ prop}}{\Gamma \vdash P \times Q \text{ prop}}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash P \text{ prop} \quad \Gamma \vdash Q \text{ prop} & \quad \Gamma \vdash s : A \quad \Gamma \vdash t : A \\
\frac{\Gamma \vdash \prod_{x:A} P \text{ prop} \quad \Gamma \vdash \text{Eq}_A(s,t) \text{ prop}}{\Gamma \vdash P \rightarrow Q \text{ prop}}
\end{align*}
\]

This still leaves room for deviant interpretations, however. For example, nothing prevents interpreting propositions as regular monos and the bracket types as epi-regular mono factorizations. For us this was not acceptable, since we wanted to obtain a calculus of bracket types that was complete for semantics in regular categories. This required specifying the properties of bracket types sufficiently to determine their interpretation uniquely.

### 3 Categorical Semantics of Bracket Types

We now sketch the semantics for bracket types in regular categories, with respect to which the rules in Figure 1 are sound and complete.

**Definition 1.** A regular category \( \mathcal{C} \) is a category with finite limits in which

1. every kernel pair has a coequalizer, and
2. the pullback of a regular epimorphism is a regular epimorphism.

Recall first how to interpret dependent type theory with dependent sums \( \sum \) and (strong, extensional) equality \( \text{Eq} \) in a category with finite limits. We use the semantic bracket \( \llbracket X \rrbracket \) to denote the interpretation
of $X$, where $X$ could be a type, a term, a context, or a judgment. When no confusion can arise, we omit the semantic brackets, especially in diagrams, to improve readability. We usually denote the interpretation of a context $x_1:A_1, \ldots, x_n:A_n$ as $(A_1, \ldots, A_n)$ instead of $[[x_1:A_1, \ldots, x_n:A_n]]$.

The empty context is interpreted as the terminal object $1$. The interpretation of a type in a context

$$\Gamma \vdash A \text{ type}$$

is given in the slice category $\mathcal{C}/[[\Gamma]]$ by an arrow, called a display map,

$$\begin{array}{c}
[[\Gamma, x:A]] \\
\downarrow \hspace{1cm} \downarrow \\
[[\Gamma]]
\end{array}$$

where we here abbreviated the name of the arrow. Its domain is the interpretation of the context $\Gamma, x:A$.

A term in a context

$$\Gamma \vdash t : A$$

is interpreted by a point of $(\Gamma, A)$ in the slice $\mathcal{C}/[[\Gamma]]$, i.e. as a section of the interpretation of $\Gamma \vdash A \text{ type}$.

The substitution of a term $a$ for a variable $x$ is interpreted as indicated in the following pullback diagram:

The interpretation of a dependent sum formed as

$$\frac{\Gamma, x:A \vdash B \text{ type}}{\Gamma \vdash \sum_{x:A} B \text{ type}}$$
is the composition of the arrows

\[
\begin{array}{c}
(I, A, B) \\
\downarrow \scriptstyle \Gamma, A \vdash B \quad \downarrow \scriptstyle \Gamma, A \vdash A \\
(\Gamma, A) \quad \downarrow \scriptstyle \Gamma \vdash \Sigma_A B \\
\downarrow \scriptstyle \Gamma \vdash A \\
(\Gamma)
\end{array}
\]

Finally, equality types are interpreted as suitable equalizers, which we shall not spell out here. This is not the place to discuss the coherence issues that arise when dependent type theory is interpreted in this way, see [Hof95] for a thorough treatment of this issue.

We now proceed with the interpretation of bracket types. A regular category \( \mathcal{C} \) has stable regular epi–mono image factorizations, meaning that every arrow \( f : A \to B \) can be factored uniquely up to isomorphism as a regular epi followed by a mono

\[
\begin{array}{c}
A \\
\downarrow f \\
\downarrow \scriptstyle \text{Im}(f) \\
B
\end{array}
\]

A bracket type

\[
\begin{array}{c}
\Gamma \vdash A \text{ type} \\
\overrightarrow{\Gamma \vdash [A] \text{ type}}
\end{array}
\]

is interpreted as the image of \( \Gamma \vdash [A] \):

\[
\begin{array}{c}
(\Gamma, A) \\
\downarrow \scriptstyle \Gamma \vdash A \\
\downarrow \scriptstyle \text{Im}(\Gamma \vdash A) \\
\text{[\Gamma, [A]]} = \text{Im}(\Gamma \vdash A)
\end{array}
\]

The regular epi part of the factorization is used in the interpretation of term bracketing

\[
\begin{array}{c}
\Gamma \vdash a : A \\
\overrightarrow{\Gamma \vdash [a] : [A]}
\end{array}
\]
The interpretation of \([a]\) is the composition

\[ (\Gamma) \xrightarrow{a} (\Gamma, A) \xrightarrow{[\cdot]} (\Gamma, [A]) \]

It remains to interpret the \texttt{where} terms. Consider

\[ \Gamma \vdash q : [A] \quad \Gamma, x : A \vdash b : B \quad \Gamma, x : A, y : A \vdash b = b[y/x] : B \]

\[ \frac{}{\Gamma \vdash b \text{ where } [x] = q : B} \]

The various terms and types occurring above are interpreted in the slice over \([\Gamma]\), as shown in the following diagram:

\[
\begin{array}{ccc}
(\Gamma, x : A, y : A) & \xrightarrow{x/y} & (\Gamma, A) \\
& \searrow^{b} & \downarrow^{\text{[\cdot]}} \\
& & (\Gamma, A, B) \\
\end{array}
\]

By assumption, the arrow labelled \(b\) coequalizes the two projections. The regular epi \([\cdot]\) is the coequalizer of those two projections, therefore \(\Gamma \vdash b\) factors uniquely through \([\cdot]\) via \(\overline{\tau}\). The interpretation of \((b \text{ where } [x] = q)\) is the composition \(\overline{b} \circ q\).

**Theorem 1.** The interpretation of bracket types in regular categories is sound and complete.

*Proof.* Soundness: The soundness of dependent sums and equality types in a category with finite limits is routine. For bracket types one needs to further verify the equality rule, two conversion rules, the substitution rules and the compatibility rules. This is also straightforward, using the existence and stability of epi-mono factorizations.

Completeness: In the standard way, one builds a syntactic category \(\mathcal{S}\) from the dependent type theory \(\mathbb{D}\) with \(1\), \(\Sigma\), \(\text{Eq}\), and \([\cdot]\) types. One then shows that \(\mathcal{S}\) is a regular category, and that the interpretation of \(\mathbb{D}\) in \(\mathcal{S}\) validates precisely all the provable equations between terms. Details can be found in [AB01].

### 4 Properties of Bracket Types

We now record without proofs some of the basic properties of bracket types. First observe that, in any context \(\Gamma\), the types satisfying proof irrelevance (1), are exactly the cartesian idempotents:

\[ P \text{ prop } \iff P = P \times_{\Gamma} P \quad (4) \]
where here and in what follows, $\equiv$ between types means that they are canonically isomorphic.

**Proposition 1.** For any types $A$, $B$ in a context $\Gamma$:

1. $[-]$ is functorial
2. There is a canonical arrow $A \to [A]$, natural in $A$
3. $[[A]] = [A]$  
4. $A = [A] \iff A = A \times_{\Gamma} A$
5. $1 = [1]$
6. $[A \times_{\Gamma} B] = [A] \times_{\Gamma} [B]$

Moreover, (1)-(4) characterize $[-]$ uniquely among stable (i.e. fibered) functors on regular categories.

It follows from the listed properties, by uniqueness of images, that one has

$$[[\sum_A B]] = [\sum_A B] .$$

For equality types $\mathsf{Eq}_A$, one has

$$[[\mathsf{Eq}_A(a, b)]] = \mathsf{Eq}_A(a, b) .$$

Together with $[1] = 1$, that already summarizes the properties of $[-]$ on its own. Things become more interesting in the presence of the other type-forming operations,

$$0, A + B, \prod_A B, A \to B, \neg A ,$$

where $\neg A$ stands for $A \to 0$.

For finite sums we get

$$[0] = 0 , \quad [A + B] = [[A] + [B]] .$$

For $\prod$ we have

$$[[\prod_A B]] = \prod_A [B] .$$

And one sees easily that

$$[[\prod_A B]] \leq \prod_A [B] .$$

Specializing to a function type $A \to B$ gives

$$[A \to [B]] = A \to [B] , \quad [A \to B] \leq A \to [B] .$$

For brackets on the left, it is easy to see that

$$A \to [B] = [A] \to [B] = [[A] \to [B]] .$$

Taking $B = 0$ therefore yields the noteworthy

$$\neg A = \neg [A] = [\neg A] .$$
5 First Order Logic via Bracket Types

In dependent type theory with the type-forming operations

\[ 0, 1, [A], A + B, \text{Eq}_A, \sum_{x:A} B, \prod_{x:A} B, \]

the propositions in every context model first-order logic, under the following definitions:

\[
\begin{align*}
\top &= 1 \\
\bot &= 0 \\
\varphi \land \psi &= \varphi \times \psi \\
\varphi \lor \psi &= [\varphi + \psi] \\
\varphi \implies \psi &= \varphi \rightarrow \psi \\
\neg \varphi &= \varphi \rightarrow \bot \\
x =_A y &= \text{Eq}_A(x, y) \\
\forall x:A, \varphi &= \prod_{x:A} \varphi \\
\exists x:A, \varphi &= [\sum_{x:A} \varphi]
\end{align*}
\]

The bracket is thus used to rectify the operations + and \( \Sigma \), which lead out of propositions. These operations will satisfy the usual rules for intuitionistic first-order logic, and the resulting system is a dependent type theory with first-order logic over each type. It can be described categorically as the internal logic of a regular tocc with finite sums. The chief difference between this formulation and more customary ones using both type theory and predicate logic is that the first-order logical operations on the propositions are here defined in terms of the operations on types, rather than taken as primitive.

The category \( \text{sh}(X) \) of sheaves on a topological space \( X \) is always a regular tocc, since it is a topos. The first-order logical operations defined as above in terms of brackets necessarily agree with the usual ones given by the topos structure. The bracket \( [E] \) of a sheaf \( E \) on \( X \) is given simply by taking the epi-mono factorization of the corresponding etale map \( E \rightarrow X \). Thus \( [E] \rightarrow X \) is just the (open) support of \( E \). We remark that by using methods similar to those used in [AB00], it can be shown that dependent type theory with the operations (9) above (and so also including first-order logic) is sound and complete with respect to such “topological semantics”, i.e., with respect to sheaf models over topological spaces.
6 First Order Logic vs. Propositions-as-Types

One can regard dependent type theory under the propositions-as-types interpretation as an intensional system of first-order logic, since two types $A$ and $B$ may be non-isomorphic and yet be logically equivalent in the sense that $A \rightarrow B$ and $B \rightarrow A$ are both inhabited. The bracket operation can then be seen as returning the “extension” $[A]$ of the “proposition in intension” $A$. Rather than pursuing this approach of a single system of intensional and extensional logic, we consider here an application of bracket types to comparing the two systems, each on its own, and without brackets. Specifically, that is, we compare conventional first-order logic with the propositions-as-types interpretation of it in dependent type theory, relating first-order provability to provability in conventional dependent type theory without brackets.

Thus suppose we have a single-sorted first-order theory $T$, consisting of constants, function and relation symbols, and axioms given as closed formulas. The standard propositions-as-types interpretation $*$ of $T$ into type theory,

$$ T \xrightarrow{*} DTT $$

is determined by fixing the interpretations of the basic sort, the constants, function and relation symbols. The rest of the interpretation is determined inductively in the evident way, using the type-forming operations in place of the corresponding logical ones, cf. [ML98]. For example,

$$(\forall x \exists y, (R(x, y) \lor P(x)))^* = \prod_{x:I^*} \sum_{y:I^*} (R^*(x, y) + P^*(x)),$$

where $I^*$ is a new basic type interpreting the domain of individuals $I$, and the dependent types $x:I^* \vdash P^*(x)$ and $x:I^*, y:I^* \vdash R^*(x, y)$ interpret the relation symbols $P$ and $R$.

If we add a constant $a : \alpha^*$ for each axiom $\alpha$, the translation $\varphi^*$ of a provable closed formula $\varphi$ becomes inhabited by a term that is obtained from a straightforward translation of the proof of $\varphi$ into type theory. Thus,

$$ \text{IFOL}(T) \vdash \varphi \implies DTT(T) \vdash \varphi^*,$$

where by $DTT(T) \vdash \varphi^*$ we mean that the type $\varphi^*$ is inhabited in the dependent type theory enriched with the basic types and constants needed for the translation $*$, and with constants inhabiting the translations of axioms of $T$. 


Now consider the converse implication: if $\varphi^*$ is inhabited in DTT($\mathbb{T}$), must $\varphi$ be provable in the intuitionistic first-order theory $\mathbb{T}$? Proofs of partial converses of (12) for different fragments of first-order logic have been given by Martin-Löf ($\forall$, $\Rightarrow$ in [ML98]) and, recently, Tait ($\exists$, $\wedge$, $\forall$, $\Rightarrow$, $\neg$ in [Tai]). These results are for type theory without equality types, and proceed from normalization. The theorem stated below applies to type theory with extensional equality and a large fragment of first-order logic.

**Definition 2.** A first-order formula $\vartheta$ is stable when it does not contain $\forall$ and $\Rightarrow$, but negation $\neg$ is allowed as a special case of $\Rightarrow$. A first-order formula $\varphi$ is left-stable when in every subformula of the form $\vartheta \Rightarrow \psi$, the formula $\vartheta$ is stable.

**Theorem 2.** If $\varphi$ is left-stable then

$$\text{DTT} = \text{IFOL} \vdash \varphi^* \text{ implies } \text{IFOL} \vdash \varphi.$$ 

Although the theorem does not involve bracket types, they are used in an essential way in the proof, to compare the two interpretations; see [AB01].

For a further application, observe that every first-order formula $\varphi$ is classically equivalent to one $\varphi^*$ that is stable. The formula $\varphi^*$, which we call the stabilized translation of $\varphi$, is obtained by replacing in $\varphi$ every $\forall x. \vartheta$ and $\vartheta \Rightarrow \psi$ by $\exists x. \neg \vartheta$ and $\neg (\vartheta \wedge \neg \psi)$, respectively. The equivalence $\varphi \iff \varphi^*$ holds intuitionistically if $\varphi = \psi^{\neg \neg}$ is the double-negation translation of a formula $\psi$. Therefore, the stabilized double-negation translation

$$(\varphi^{\neg \neg})^s$$

takes a formula $\varphi$ of classical first-order logic (CFOL) to a stable one in IFOL, with the property

$$\text{CFOL} \vdash \varphi \text{ if, and only if, } \text{IFOL} \vdash (\varphi^{\neg \neg})^s$$

If we compose the $\neg \neg$ translation with the propositions-as-types translation $\ast$, we obtain a translation

$$\varphi^+ = ((\varphi^{\neg \neg})^s)^\ast$$

which takes formulas of classical first-order logic into dependent type theory.
Corollary 1. The translation $\varphi \mapsto \varphi^+$ of classical first-order logic into dependent type theory has the following property:

$$\text{CFOL} \vdash \varphi \text{ if, and only if, } \text{DTT} \vdash \varphi^+$$

Here DTT $\vdash \varphi^+$ means that the type $\varphi^+$ is inhabited.

Remark 1. The following formula was suggested to us by Thierry Coquand:

$$(\forall x \exists y. R(x, y)) \implies \forall x, x'. \exists y, y'. (R(x, y) \land R(x', y') \land (x = x' \implies y = y')).$$

It is not provable in intuitionistic first-order logic, but its *-translation is inhabited in dependent type theory. Theorem 2 therefore cannot be extended to full intuitionistic first-order logic.

Remark 2. For the special case of intuitionistic propositional logic (IPC, with connectives $\top, \land, \implies, \bot, \lor$) completeness with respect to dependent type theory (DTT) is easily seen to hold for all formulas. Briefly, first we use the Curry-Howard correspondence between proofs in IPC and terms in simply-typed $\lambda$-calculus with disjoint sums and the empty type (STT) to conclude that

$$\text{IPC} \vdash \varphi \text{ if, and only if } \text{STT} \vdash \varphi^*.$$

Then we use the well-known correspondence between STT and bicartesian closed categories (BiCCC),\(^1\) to conclude that the completeness of IPC with respect to DTT follows from the fact that every BiCCC has a full and faithful BiCCC embedding into a locally cartesian closed category with stable finite coproducts.

Acknowledgements. We gratefully acknowledge the support of the Institut Mittag-Leffler, the Royal Swedish Academy of Sciences, where this research was conducted (cf. [AB01]). We thank Peter Aczel, Lars Birkedal, Thierry Coquand, Nicola Gambino, Milli Maietti, Per Martin-Löf, Grigori Mints, Erik Palmgren, Frank Pfenning, Dana Scott, Anton Setzer, and William Tait for their contributions. This work is part of the Logic of Types and Computation Project at Carnegie Mellon University under the direction of Dana Scott [ca02].

\(^1\) Finite coproducts are required to be stable under pullbacks here, otherwise substitution rules are unsound.
A Dependent Sums and Equality Types

For the sake of completeness, we list the rules for dependent type theory with the unit type, strong dependent sums and strong extensional equality, cf. [Jac99]. We do not show rules which relate judgmental equality and substitution ("substitution of equals for equals").

Formation rules:

\[ \frac{}{\Gamma \vdash \text{1 type}} \]
\[ \frac{\Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \sum_{x : A} B \text{ type}} \]
\[ \frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A, y : A \vdash \text{Eq}_A(x, y) \text{ type}} \]

Introduction and elimination rules:

\[ \frac{}{\Gamma \vdash * : 1} \]
\[ \frac{\Gamma \vdash a : A}{\Gamma \vdash \langle a, b \rangle : \sum_{x : A} B} \]
\[ \frac{\Gamma \vdash p : \sum_{x : A} B}{\Gamma \vdash \pi_1(p) : A} \]
\[ \frac{\Gamma \vdash p : \sum_{x : A} B}{\Gamma \vdash \pi_2(p) : B(\pi_1(p)/x)} \]
\[ \frac{\Gamma \vdash t : A}{\Gamma \vdash r(t) : \text{Eq}_A(t, t)} \]

Equality rules:

\[ \frac{}{\Gamma \vdash t : 1} \]
\[ \frac{\Gamma \vdash e : \text{Eq}_A(s, t)}{\Gamma \vdash s = t : A} \]
\[ \frac{\Gamma \vdash e : \text{Eq}_A(s, t)}{\Gamma \vdash e = r(s) : \text{Eq}_A(s, t)} \]

Conversions:

\[ \pi_1(\langle a, b \rangle) = a \]
\[ \pi_2(\langle a, b \rangle) = b \]
\[ \langle \pi_1(p), \pi_2(p) \rangle = p \]
References


