# Using Signatures in Type Theory to Represent Situations*

Stergios Chatzikyriakidis and Zhaohui Luo

Department of Computer Science
Royal Holloway, University of London
stergios.chatzikyriakidis@cs.rhul.ac.uk
zhaohui.luo@hotmail.co.uk

**Abstract.** Signatures have been introduced to represent situations in the formal semantics based on modern type theories. In this paper, we study the notion of signature in more details, presenting it formally and discussing its use in representations of situations. In particular, the new forms of signature entries, the subtyping entries and the manifest entries, are formally studied and the corresponding extensions are shown to enjoy the nice meta-theoretical properties. Besides being signature entries, these two forms of entries may be introduced to form contextual entries as well and this may have interesting implications in applications of the notion of context to, for example, belief contexts.

## 1 Introduction

Signatures are introduced to represent situations (or incomplete possible worlds) by the second author in [13], where it has been argued that, with the new forms of subtyping entries and manifest entries, signatures are very useful in representing situations in a formal semantics based on modern type theories (MTTs). In this paper, we shall study the notion of signature in a more formal and detailed way.

The notion of signature has been used in describing algebraic structures. Its use in type theory can be found in the Edinburgh Logical Framework [7].There, signatures are used to describe constants (and their types) in a logical system. This is in contrast to contexts in type theory that describe variables (and their types) which can be abstracted by means of quantification or $\lambda$-abstraction. We shall study the notion of signature in MTTs by extending the logical framework LF (Chapter 9 of [8]) with signatures to obtain the system $\mathrm{LF}_\Sigma$, which can be used similarly as LF in specifying type theories such as Martin-Löf's type theory [16] and UTT [8].

Signatures as proposed in [13] may contain two new forms of entries: subtyping entries and manifest entries. A subtyping entry $A <_\kappa B$ declares that $A$ is a subtype of $B$ via. coercion $\kappa$. This localises a coercive subtyping relationship

* This work is partially supported by the research grant F/07-537/AJ of the Leverhulme Trust in U.K. and the CAS/SAFEA International Partnership Program for Creative Research Teams.

as studied in the coercive subtyping framework [9,15] that was developed for type theory based proof assistants. Subtyping has been proved useful in formal semantics and, specifically for MTT-semantics, it is crucial partly because CNs are interpreted as types rather than predicates (as in Montague semantics). It is very useful to introduce subtyping entries in signatures when they are used to represent situations. Also, we shall explain that the introduction of coherent subtyping entries to signatures preserves the nice properties of the original type theory.

The other new form of signature entries is that of manifest entries. These have the form $c \sim a : A$, which introduces the constant $c$ and assumes that it behaves exactly like the object $a$ of type $A$.[1] Formally, a manifest entry is just the abbreviation of an ordinary membership entry together with a subtyping entry. The latter enforces the abbreviation: $c \sim a : A$ abbreviates $c : \mathbf{1}_A(a)$, where $\mathbf{1}_A(a)$ is the inductive unit type, together with the subtyping entry $\mathbf{1}_A(a) <_{\xi_{A,a}} A$ with the coercion $\xi_{A,a}$ that maps the object of the unit type to $a$. Such an extension with manifest entries is sound: meta-theoretically, the extension preserves all of the nice properties of the original type theory. We shall make this clear in more detail in the paper.

Both subtyping and manifest entries can be considered as contextual entries for declaring variables. This makes contributions to the application of contexts. One such example can be found in Ranta's treatment of belief contexts [19]. We show how to extend this notion of belief context with these new entries and how they allow belief contexts to express infinitely many beliefs among others. We shall also point out that, if we introduce these to form contextual entries, we should allow the corresponding move to the right of the turnstile: by quantification and $\lambda$-abstraction for manifest entries and by local coercions for subtyping entries. In particular, for subtyping entries, this requires the introduction of the new form of terms, **coercion** $A <_c B$ **in** $M$, to express local coercions[2] and this may make the meta-theoretical study more sophisticated.

The notion of signature is formally introduced in §2, where we present the system $\mathrm{LF}_\Sigma$ and give an example to illustrate its use in representations of situations. In §3, the subtyping and manifest entries in signatures are studied: they are shown to be useful in expressing situations and the extensions with them preserve the nice meta-theoretic properties. The potential of adding such new forms of entries as contextual entries is considered in §4, where we use belief contexts as an example to illustrate that this can be useful.

[1] Contextual manifest entries were first proposed by the second author in [10], where they are studied in a different context, focussing on its intensional nature, as compared with traditional extensional definition entries in proof assistants.

[2] Local coercions are useful in formal semantics based on MTTs. See, for example, [12] for discussions.

## 2 Signatures for Representing Situations

Situations, or incomplete possible worlds, are proposed to be represented by *signatures* in MTT-semantics, i.e., when modern type theories are used to give formal semantics [13].

The use of possible worlds in set theory has been a central mechanism within Montagovian approaches of formal semantics, especially, in dealing with intensional phenomena including, for example, belief intensionality among other things. However, the use of set-theoretical possible worlds has given rise to the well-known hyperintensional problem, with various paradoxes associated with it (e.g., the Paris Hilton paradox and the woodchuck-groundhog paradox) [18]. When intensional type theories are employed for formal semantics, types rather sets are used to interpret CNs and significantly different mechanisms are available in representing and dealing with such phenomena. Using signatures to represent situations is such a proposal.

We shall describe the notion of signature formally, compare it with that of context, and give a simple example of its use in representing situations. In this section. we shall only describe signatures with the traditional membership entries. Contexts with such traditional entries have been used by Ranta [19] and others [3,6] to represent situations, where they do not consider the issue of difference between variables and constants. We consider signatures rather than contexts here. Note that signatures may contain other forms of entries which are studied in the next section §3.

### 2.1 Signatures in Type Theory: a Formal Presentation

Type theories can be specified in a logical framework such as Martin-Löf's logical framework [16] or its typed version LF [8]. We shall extend LF with signatures to obtain $\mathrm{LF}_{\Sigma}$.

Informally, a signature is a sequence of entries of several forms, one of which is the form of membership entries $c : K$, which is the traditional form of entries as occurred in contexts (we shall add two other forms of entries in the next section). If a signature has only membership entries, it is of the form $c_1 : K_1, ..., c_n : K_n$.

LF is a dependent type theory whose types are called *kinds* in order to be distinguished from types in the object type theory. It has the kind $Type$ of all types of the object type theory and dependent $\Pi$-kinds of the form $(x{:}K)K'$ (we omit their details here – see [8]). In LF, there are five forms of judgements:

- $\vdash \Gamma$ (or written as '$\Gamma$ *valid*'), which asserts that $\Gamma$ is a valid context.
- $\Gamma \vdash K$ *kind*, which asserts that $K$ is a kind in $\Gamma$.
- $\Gamma \vdash k : K$, which asserts that $k$ is an object of kind $K$ in $\Gamma$.
- $\Gamma \vdash K_1 = K_2$, which asserts that $K_1$ and $K_2$ are equal kinds in $\Gamma$.
- $\Gamma \vdash k_1 = k_2 : K$, which asserts that $k_1$ and $k_2$ are equal objects of kind $K$ in $\Gamma$.

To extend LF with signatures, we amend each form of judgement with a signature $\Sigma$ and add another form of judgements saying that a signature is valid. In other words, $\mathrm{LF}_{\Sigma}$ has the following six forms of judgements:

**Signature Validity and Assumptions**

$$\frac{}{\langle\rangle\ valid} \qquad \frac{\langle\rangle \vdash_\Sigma K\ kind \quad c \notin dom(\Sigma)}{\Sigma,\ c : K\ valid} \qquad \frac{\vdash_{\Sigma,c:K,\Sigma'} \Gamma}{\Gamma \vdash_{\Sigma,c:K,\Sigma'} c : K}$$

**Context Validity and Assumptions**

$$\frac{\Sigma\ valid}{\vdash_\Sigma \langle\rangle} \qquad \frac{\Gamma \vdash_\Sigma K\ kind \quad x \notin dom(\Gamma)}{\vdash_\Sigma \Gamma,\ x : K} \qquad \frac{\vdash_\Sigma \Gamma, x : K, \Gamma'}{\Gamma, x : K, \Gamma' \vdash_\Sigma x : K}$$

**Fig. 1.** Rules for signatures/contexts in $\mathrm{LF}_\Sigma$.

- $\Sigma\ valid$, which asserts that $\Sigma$ is a valid signature.
- $\vdash_\Sigma \Gamma$, which asserts that $\Gamma$ is a valid context under $\Sigma$.
- $\Gamma \vdash_\Sigma K\ kind$, which asserts that $K$ is a kind in $\Gamma$ under $\Sigma$.
- $\Gamma \vdash_\Sigma k : K$, which asserts that $k$ is an object of kind $K$ in $\Gamma$ under $\Sigma$.
- $\Gamma \vdash_\Sigma K_1 = K_2$, which asserts that $K_1$ and $K_2$ are equal kinds in $\Gamma$ under $\Sigma$.
- $\Gamma \vdash_\Sigma k_1 = k_2 : K$, which asserts that $k_1$ and $k_2$ are equal objects of kind $K$ in $\Gamma$ under $\Sigma$.

All of the inference rules of LF (those in Figures 9.1 and 9.2 of Chapter 9 of [8]) become inference rules of $\mathrm{LF}_\Sigma$ after replacing $\vdash$ by $\vdash_\Sigma$ (and changing the judgement form '$\Gamma\ valid$' to '$\vdash_\Sigma \Gamma$'). For instance, the following rule for $\lambda$-abstraction[3] in LF

$$\frac{\Gamma, x : K \vdash b : K'}{\Gamma \vdash [x{:}K]b : (x{:}K)K'}$$

becomes, in $\mathrm{LF}_\Sigma$,

$$\frac{\Gamma, x : K \vdash_\Sigma b : K'}{\Gamma \vdash_\Sigma [x{:}K]b : (x{:}K)K'}$$

In addition, in $\mathrm{LF}_\Sigma$, we have the rules in Figure 1 for signatures (and contexts), concerning their validity and their roles of making basic assumptions, where $\langle\rangle$ is the empty sequence and $dom(p_1 : K_1,\ ...\ p_n : K_n) = \{p_1, ..., p_n\}$. Note that the assumptions in a signature or in a context can be derived – this is characterised by the third rule and the last rule in Figure 1, respectively.

*Remark 1.* The membership entry $c : K$ in a signature declares that $c$ is a *constant* of kind $K$. This is different from a contextual entry $x : K$ that declares $x$ to be a *variable*. Note that a variable can be abstracted by, for example, quantification or $\lambda$-abstraction as exemplified by a rule like the one below, where $Prop$ is the universe of logical propositions:

$$\frac{\Gamma,\ x : K \vdash_\Sigma P : Prop}{\Gamma \vdash_\Sigma \forall x{:}K.P : Prop}$$

[3] In LF, we use the notation $[x{:}K]b$ for $\lambda x{:}K.b$ and $(x{:}K)K'$ for $\Pi x{:}K.K'$.

However, constants in signatures can never be abstracted in this way – that is why they are called constants. Therefore, signatures can adequately be used to represent situations. Also, because the constants in signatures cannot be abstracted, it is easier meta-theoretically to add new forms of entries to signatures than to contexts (see later).

### 2.2 Use of Signatures to Represent Situations: a Simple Example

Signatures can adequately be used to represent situations, or incomplete possible worlds, in the MTT-semantics. This possibility can easily be understood whenever one realises that types represent collections of objects just like sets, although types are syntactic (or, better, proof-theoretic) entities different from sets in set theory. Intuitively, the similarity between types and sets is one of the crucial reasons that MTT-semantics can be viewed as model-theoretic, while the differences between types and sets and, especially that the former are proof-theoretically defined, are why MTT-semantics can be also viewed as proof-theoretic (see [13] for more details).

That signatures can be used to represent situations is the other facet that the MTT-semantics is model-theoretic. Here, we use an example given in [13] to illustrate how signatures can be used to represent situations.

*Example 1.* The example, taken from Chapter 10 of [20], is about an (imagined) situation in the Cavern Club at Liverpool in 1962 where the Beatles were rehearsing for a performance. This situation can be represented as follows.

1. The domain of the situation consists of several peoples including the Beatles (John, Paul, George and Ringo), their manager (Brian) and a fan (Bob). This can be represented be means of the following signature $\Sigma_1$:

$$\Sigma_1 \equiv D : Type,$$
$$John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$$

2. The assignment function assigns, for example, predicate symbols such as $B$ and $G$ to the propositional functions expressing 'was a Beatle' and 'played guitar', respectively. We can introduce the following in our signature to represent such an assignment function:

$$\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$$
$$G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$$

The signature that represents the situation will be of the form $\Sigma \equiv \Sigma_1,\ \Sigma_2,\ ...,\ \Sigma_n$. We shall then have, for instance,

$$\vdash_\Sigma G(John)\ \ true\ \ and\ \ \vdash_\Sigma \neg B(Bob)\ \ true.$$

where $G(John)$ and $B(Bob)$ are the semantic interpretations of `John played Guitar` and `Bob was a Beatle`, respectively.

$$\frac{\vdash_\Sigma A : Type \quad \vdash_\Sigma B : Type \quad \vdash_\Sigma c : (A)B}{\Sigma, A <_c B \ valid} \qquad \frac{\vdash_{\Sigma, A <_c B, \Sigma'} \Gamma}{\Gamma \vdash_{\Sigma, A <_c B, \Sigma'} A <_c B}$$

**Fig. 2.** Rules for subtyping entries in signatures.

## 3 Subtyping and Manifest Entries in Signatures

In the last section, we introduced signatures with only traditional membership entries. In this section, we consider two other forms of entries – the subtyping entries and manifest entries: introducing them into signatures, discussing meta-theoretic implications and illustrating their uses in representing situations.

In earlier work, these forms of entries were considered contextual entries: contextual manifest entries were first studied in [10] and contextual subtyping entries (in so-called coercion contexts) in [12]. Here in this section, we consider them as entries in signatures, as proposed in [13]. For this reason, they are not only useful in representing situations, but are also simpler meta-theoretically, since they are introducing constants rather than variables and, as a consequence of the subtyping entries, one does not need to introduce corresponding terms for the purpose of making abstraction operations possible (see §4 for further discussion in this last respect).

### 3.1 Subtyping Entries and Their Uses

Coercive subtyping has been studied for subtyping and abbreviations in MTTs and the associated proof assistants [9,15].[4] Introducing subtyping entries (to either signatures or contexts) is to localise the coercive subtyping mechanism, which has been studied globally in earlier research.

Syntactically, the system $\mathrm{LF}_\Sigma$ is extended with the judgement forms $\Gamma \vdash_\Sigma A <_\kappa B : Type$ (we shall often just write $A <_\kappa B$ even when $A$ and $B$ are types) and $\Gamma \vdash_\Sigma K <_\kappa K'$. A subtyping entry to signatures can be introduced by means of the first rule in Figure 2, where $(A)B$ is the kind of functional operations from $A$ to $B$. The second rule in Figure 2 expresses that the subtyping assumptions in a signature are derivable. Then the rules for coercive subtyping [15], albeit extended for judgements with signatures, are all applicable. For instance, if signature $\Sigma$ contains $A <_\kappa B$ and $B <_{\kappa'} C$, we can derive $A <_{\kappa' \circ \kappa} C$ under $\Sigma$.

It is worth pointing out that *validity* of a signature is not enough anymore when we consider subtyping entries in signatures. For signature $\Sigma$ to be legal,

[4] The word 'coercion' has been used for related but maybe different things including coercions in programming languages and coercions in linguistics. See Asher and Luo [1] for a use of coercive subtyping in modelling linguistic coercions and Retoré et al [2] for another proposal of using coercions to deal with some linguistic coercions in lexical semantics.

we need the subtyping assumptions in $\Sigma$ to be *coherent* in the sense that, informally, all coercions between any two types are equal, i.e., in some appropriate subsystem,[5] if $\Gamma \vdash_\Sigma A <_\kappa B$ and $\Gamma \vdash_\Sigma A <_{\kappa'} B$, then $\Gamma \vdash_\Sigma \kappa = \kappa' : (A)B$.

We can then show that the conservativity result in [15] can be carried over to the current setting and, in particular, if the original type theory is strongly normalising, so is the type theory extended with the subtyping entries. As a consequence, the extension with subtyping entries preserves logical consistency – a basic requirement for a type theory to be employed for formal semantics.

Introducing subtyping entries makes using type theory for formal semantics much more convenient. First of all, it is now possible for one to *localise* subtyping assumptions. In some specific situations, some special subtyping relations may reasonably be assumed, which may not be reasonable in general. For instance, only in a cafe or restaurant would it be reasonable to say

(1) The ham-sandwich left without paying the bill.

In representing a situation in a cafe, we might reasonably assume the following subtyping entry:

$$Ham\text{-}sandwich < Human,$$

which will then allow the sentence (1) to be semantically interpreted as intended. Such reference transfers are studied by Nunberg [17] among others.

### 3.2 Manifest Entries and Their Uses

A *manifest entry* is of the form

$$c \sim a : A \tag{2}$$

Informally, it assumes that $c$ behaves exactly like $a$ of type $A$. Alternatively, one can think that in any place that we could use an object of type $A$, we could use $c$ which actually plays the role of $a$. Signatures can be extended with manifest entries:

$$(*) \qquad \frac{\vdash_\Sigma A : Type \quad \vdash_\Sigma a : A \quad c \notin dom(\Sigma)}{\Sigma,\ c \sim a : A\ valid}$$

where $Type$ is the kind of all types (in the object type theory). In fact, such manifest entries can be introduced by means of special membership entries with the help of the coercive subtyping mechanism. We now proceed with its formal description.

Manifest entries can be regarded as abbreviations of special membership entries [10] with the help of the coercive subtyping mechanism [9,15]. Formally,

[5] It is important that the condition is not stated for the whole system of coercive subtyping, for otherwise it would become trivial. Here we do not detail the description of the subsystem because we would then have to make explicit some technical details we feel unnecessary for this paper. An interested reader may look at [15] for details how coherence is defined in a global case.

to add the above manifest entry (2) to a signature is to add the following two entries:

$$c : \mathbf{1}_A(a), \quad \mathbf{1}_A(a) <_{\xi_{A,a}} A \tag{3}$$

where $\mathbf{1}_A(a)$ is the inductive unit type parameterised by $A : Type$ and $a : A$, whose only object is $*_A(a)$, and $\xi_{A,a}(x) = a$ for every $x : \mathbf{1}_A(a)$. It is now easy to see that, if an expression has a hole that requires a term of type $A$, we can use $c$ to fill that hole; then the whole expression is equal to that with the hole filled by $a$. For example, if the expression is $f(_)$, then $f(c)$ is equal to $f(a)$.

Note that the subtyping entries involving $\xi$ form coherent signatures; in particular, if for two manifest entries $c \sim a : A$ and $d \sim b : B$ we have $\mathbf{1}_A(a) = \mathbf{1}_B(b)$ and $A = B$, then $\xi_{A,a} = \xi_{B,b}$, as coherence requires. Put in another way, if the subtyping entries in a signature are coherent, the signature is coherent since its manifest entries do not cause incoherence. Therefore, the extension with manifest entries in signatures preserves the nice properties of the original type theory such as strong normalisation and logical consistency.

Manifest entries can considerably reduce the complexity of representation, as the following example shows.

*Example 2.* With manifest entries, the situation in Example 1 can be represented as the following signature:

$$D \sim a_D : Type,\ B \sim a_B : D \to Prop,\ G \sim a_G : D \to Prop,\ \ldots\ \ldots \tag{4}$$

where

- $a_D = \{John,\ Paul,\ George,\ Ringo,\ Brian,\ Bob\}$ is a finite type,
- $a_B : D \to Prop$, the predicate 'was a Beatle', is an inductively defined function such that $a_B(John) = a_B(Paul) = a_B(George) = a_B(Ringo) = True$ and $a_B(Brian) = a_B(Bob) = False$, and
- $a_G : D \to Prop$, the predicate 'played guitar', is an inductively defined function such that $a_G(John) = a_G(Paul) = a_G(George) = True$ and $a_G(Ringo) = a_G(Brian) = a_G(Bob) = False$.

In other words, $\Sigma_1$ in Example 1 is now expressed by the first entry of (4) and $\Sigma_2$ in Example 1 by the second and third entries of (4).

Manifest entries in signatures can be used to represent infinite situations such as those with infinite domains. With traditional membership entries (as in the traditional notion of context), we can only describe finite domains as we have done in Example 1. What if the domain $D$ is infinite? This can be done by using a manifest entry – as in Example 2, we can assume that

$$D \sim Inf : Type,$$

where $Inf$ is some inductively defined type with infinitely many objects. Similarly, one can assume an infinite predicate over the domain, represented as:

$$P \sim \text{P-defn} : D \to Prop,$$

where P-defn is also inductively defined.

## 4 Subtyping and Manifest Entries in Contexts

The subtyping or manifest entries may be introduced in contexts as well. If this were done, it would further widen the uses of contexts in their applications. However, before introducing them and illustrating their uses be means of belief contexts, we should make clear that introducing contextual subtyping entries (and manifest entries, which have associated subtyping entries via $\xi$) complicates meta-theoretic studies. Until now, although the proposal of introducing contextual subtyping entries was already made in 2009 [11], the corresponding meta-theoretic studies have not been carried out in detail (for an initial study of this, see [14]) and further studies are needed.

Because a contextual entry should be able to be abstracted or moved to the right of turnstile (see Remark 1), it is necessary to introduce a new form of terms so that subtyping assumptions in a context can be represented as *local coercions* in terms. An term with a local coercion is of the form

$$\textbf{coercion } A <_\kappa B \textbf{ in } M,$$

which indicates that the scope in which subtyping $A <_\kappa B$ takes effects is term $M$ – it does not take effect outside $M$. Local coercions are introduced the rules like the following:

$$\frac{\Gamma, A <_\kappa B \vdash_\Sigma k : K}{\Gamma \vdash_\Sigma (\textbf{coercion } A <_\kappa B \textbf{ in } k) : (\textbf{coercion } A <_\kappa B \textbf{ in } K)}$$

where the parentheses are there for readability, but not necessary.

Ranta [19] has proposed an account of belief intensionality in which he uses contexts to model agents' beliefs as a sequence of membership entries.[6] The idea is simple and it is based on the assumption that contexts can be seen as the equivalent type theoretic notion of a (partial) world as found in the traditional Montagovian semantics. Ranta introduces an agent's belief context: for agent $p$, $p$'s belief context may be:

$$\Gamma_p = x_1 : A_1, ..., x_n : A_n.$$

A belief operator is then introduced: for a proposition $A$, $B_p(A)$ is true just in case that $A$ is true in $p$'s belief context $\Gamma_p$, which is equivalent to saying that $\Pi x_1{:}A_1 ... \Pi x_n{:}A_n.A$ is true.[7]

In a case like (5):

(5) John believes that all woodchucks are woodchucks ⇒ John believes all woodchucks are groundhogs.

[6] Similar ideas have been put forth in [5] and [4] to deal with intensional adjectives and adverbs.

[7] Here, we do not discuss the issue whether such a proposal is adequate to represent intensional beliefs. For instance, one might argue against such proposals simply by arguing that ordinary logical inference does not capture the intended inference concerning beliefs. We are simply take this as an example to show that the usefulness of subtyping/manifest entries in contexts.

the sentences are evaluated against the agent's belief context. If, from John's belief context, one cannot derive the belief that 'all woodchucks are groundhogs',[8] the unwanted entailment (5) does not go through.

If we introduce subtyping entries and manifest entries into contexts, we would then be able to make the above mechanism for beliefs more powerful. Here are some examples:

- In one's belief context, there can be subtyping entries like $Man < Human$ (or even unreasonably $Human < Man$).
- Infinite beliefs can be expressed by manifest entries. In particular, we can use inductive definitions to capture infinitely many entries by means of finitely many entries.

Formally, when contexts are extended with subtyping (and manifest) entries, the belief operator $B_p(P)$ can be defined as follows.

**Definition 1** *First, define $B_\Gamma$ for arbitrary context $\Gamma$ as follows.*

1. *If $\Gamma = \langle\rangle$, then $B_\Gamma(P) = P$.*
2. *If $\Gamma = x : A, \Gamma_0$, then $B_\Gamma(P) = \Pi x{:}A.\ B_{\Gamma_0}(P)$.*
3. *If $\Gamma = A <_\kappa B, \Gamma_0$, then $B_\Gamma(P) =$ **coercion** $A <_\kappa B$ **in** $B_{\Gamma_0}(P)$.*

*Then, let $p$ be an agent and $P$ a $\Gamma_p$-proposition. Define the belief operator as*

$$B_p(P) = B_{\Gamma_p}(P).$$

*Remark 2.* In the above definition, we have not considered manifest entries because a manifest entry can be represented by an ordinary membership entry together with a subtyping entry and, therefore, the above definition covers manifest entries as well.

## References


1. N. Asher and Z. Luo. Formalisation of coercions in lexical semantics. *Sinn und Bedeutung 17, Paris*, 2012.
2. C. Bassac, B. Mery, and C. Retoré. Towards a type-theoretical account of lexical semantics. *Journal of Logic, Language and Information*, 19(2), 2010.
3. P. Boldini. Formalizing contexts in intuitionistic type theory. *Fundamenta Informaticae*, 4(2), 2000.
4. S. Chatzikyriakidis. Adverbs in a modern type theory. In *LACL 2014*, 2014.
5. S. Chatzikyriakidis and Z. Luo. Adjectives in a modern type-theoretical setting. In G. Morrill and J.M Nederhof, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pages 159–174, 2013.
6. R. Dapoigny and P. Barlatier. Modelling contexts with dependent types. *Fundamenta Informaticae*, 104, 2010.


[8] For example, using the heterogenous equality $Eq$, this belief can be expressed as $\forall x{:}G \forall y{:}W.Eq(G, W, x, y)$. We do not get into the formal details here.

7. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.
8. Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.
9. Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.
10. Z. Luo. Manifest fields and module mechanisms in intensional type theory. In S. Berardi, F. Damiani, and U. de'Liguoro, editors, *Types for Proofs and Programs, Proc. of Inter. Conf. of TYPES'08, LNCS 5497.*, 2009.
11. Z. Luo. Type-theoretical semantics with coercive subtyping. *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, 2010.
12. Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.
13. Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? *Invited talk at Logical Aspects of Computational Linguistics 2014 (LACL 2014), Toulouse. LNCS 8535*, pages 177–188, 2014.
14. Z. Luo and F. Part. Subtyping in type theory: Coercion contexts and local coercions (extended abstract). *TYPES 2013, Toulouse*, 2013.
15. Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 223:18–42, 2012.
16. B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990.
17. Geoffrey Nunberg. Transfers of meaning. *Journal of Semantics*, 12(2):109–132, 1995.
18. C. Pollard. Hyperintensions. *Journal of Logic and Computation*, 18(2), 2008.
19. A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
20. J. Saeed. *Semantics*. Wiley-Blackwell, 1997.