# Dependent Event Types

Zhaohui Luo[1]* and Sergei Soloviev[2]**

[1] Royal Holloway, University of London, U.K.
zhaohui.luo@hotmail.co.uk
[2] IRIT, Toulouse, France
Sergei.Soloviev@irit.fr

**Abstract.** This paper studies how dependent types can be employed for a refined treatment of event types, offering a nice improvement to Davidson's event semantics. We consider dependent event types indexed by thematic roles and illustrate how, in the presence of refined event types, subtyping plays an essential role in semantic interpretations. We consider two extensions with dependent event types: first, the extension of Church's simple type theory as employed in Montague semantics that is familiar with many linguistic semanticists and, secondly, the extension of a modern type theory as employed in MTT-semantics. The former uses subsumptive subtyping, while the latter uses coercive subtyping, to capture the subtyping relationships between dependent event types. Both of these extensions have nice meta-theoretic properties such as normalisation and logical consistency; in particular, we shall show that the former can be faithfully embedded into the latter and hence has expected meta-theoretic properties. As an example of applications, it is shown that dependent event types give a natural solution to the incompatibility problem (sometimes called the event quantification problem) in combining event semantics with the traditional compositional semantics, both in the Montagovian setting with the simple type theory and in the setting of MTT-semantics.

## 1 Introduction

The event semantics, whose study was initiated by Davidson [6] and further studied in its neo-Davidsonian turn (see [17] among others), has several notable advantages including Davidson's original motive to provide a satisfactory semantics for adverbial modifications. Dependent types, as those found in Modern Type Theories such as Martin-Löf's type theory [15] and UTT [11], provide a useful tool in formalising event types and a nice treatment of the event semantics.

In this paper, we shall study event types that may depend on thematic roles such as agents and patients of the events. For example, we can consider the type

* Partially supported by EU COST Action CA15123 and CAS/SAFEA International Partnership Program.

** Also an associated researcher at ITMO Univ, St. Petersburg, Russia. Partially supported by EU COST Action CA15123 and Russian Federation Grant 074-U01.

$Evt_{AP}(a, p)$ of events whose agent and patient are $a$ and $p$, respectively. We shall investigate subtyping relations between event types which include dependent types such as $Evt_{AP}(a, p)$ and the non-dependent type $Event$ of all events (the latter is found in the traditional setting). For example, it may be natural to have $Evt_{AP}(a, p) \leq Evt_A(a)$, that is, the type of events with agent $a$ and patient $p$ is a subtype of that with agent $a$. With such subtyping relations in place, the semantics of verb phrases can now take the usual non-dependent types, as in the traditional setting, although dependent event types are considered.

It is shown that such dependent event types give a natural solution to the incompatibility problem in combining event semantics with the traditional Montague semantics [2, 20] (sometimes called the event quantification problem [7]). When introducing events into formal semantics, one faces a problem, which is long-standing and has seemed intractable: it comes from the issue of scopes for two kinds of quantifiers – the existential quantifier over an event variable and the other quantifiers such as one that arises from a quantificational noun phrase (see §5 for examples). It is in general expected that the correct semantics is obtained when the event quantifier takes the lower scope, but the problem is that, even when the event quantifier takes a wider scope, which would give an incorrect semantics, the resulting semantic formula is still well-formed formally. This has led to many proposals such as that considered by Champollion [2] or the related Scope Domain Principle proposed by Landman [9], but all of them are rather ad hoc. Dependent event types will solve this problem: they give a solution where the correct semantics are accepted while the incorrect ones are excluded by typing because they would be ill-typed and hence formally illegal.

Dependent event types (DETs) were first considered in an example in [1] to study linguistic coercions in formal semantics, where types of events are indexed by their agents: $Evt(h)$ is the type of events conducted by $h : Human$. In this paper, we shall study event types dependent on thematic roles in event semantics both in the traditional Montague semantics [16] and in formal semantics in modern type theories (MTT-semantics, for short) [18, 13, 4]. For the former, we extend Church's simple type theory [5], as employed in Montague semantics that is familiar with many linguistic semanticists, by means of dependent event types, resulting in the system $C_e$, where the subtyping relationships between DETs are captured by subsumptive subtyping. For the latter, we extend an MTT (in particular, the type theory UTT [11]) with DETs whose subtyping relationships are reflected by coercive subtyping [12, 14], resulting in the type system UTT[E]. Both of these extensions have nice meta-theoretic properties such as normalisation and logical consistency; in particular, we shall show that $C_e$ can be faithfully embedded into UTT[E] and hence has desirable properties.

The rest of the paper is organised as follows. In §2, we shall describe the basics of DETs, introducing notations and examples. Subtyping between event types is described in §3, where we show, for example, how VPs can take the traditional non-dependent type, while we consider DETs. The formal systems $C_e$ and $UTT[E]$ and the embedding of $C_e$ in UTT[E] are studied in §4. §5 considers the solution of the event quantification problem by means of DETs: §5.1 shows

examples in the Montagovian setting and §5.2 considers it in MTT-semantics. The concluding section briefly discusses the future work.

## 2 Dependent Event Types

In the Davidsonian event semantics in the traditional Montagovian setting [6, 17], there is only one type *Event* of all events. For example, the sentence (1) is interpreted as (2):

(1) John kissed Mary passionately.

(2) $\exists e : \mathit{Event}.\ \mathit{kiss}(e)\ \&\ \mathit{agent}(e, j)\ \&\ \mathit{patient}(e, m)\ \&\ \mathit{passionate}(e)$

where in (2), *Event* is the type of all events, *kiss*, *passionate* : $\mathit{Event} \to \mathbf{t}$ are predicates over events, and *agent*, *patient* : $\mathit{Event} \to \mathbf{e} \to \mathbf{t}$ are relations between events and entities.[3] Please note that, in the above neo-Davidson's semantics (2), adverbial modifications and thematic role relations are all propositional conjuncts in parallel with the verb description, an advantageous point as compared with an interpretation without events.

We propose to consider refined types of events. Rather than a single type *Event* of events, we introduce types of events that are dependent on some parameters. For instance, an event type can be dependent on agents and patients. Let *Agent* and *Patient* be the types of agents and patients, respectively. Then, for $a : \mathit{Agent}$ and $p : \mathit{Patient}$, the dependent type

$$\mathit{Evt}_{AP}(a, p)$$

is the type of events whose agents are $a$ and whose patients are $p$. With such dependent event types, the above sentence (1) can now be interpreted as:[4]

(3) $\exists e : \mathit{Evt}_{AP}(j, m).\ \mathit{kiss}(e)\ \&\ \mathit{passionate}(e)$

Note that, besides other things we are going to explain below, we do not need to consider the relations *agent* and *patient* as found in (2) because they can now be 'recovered' from typing. For example, for $a : \mathit{Agent}$ and $p : \mathit{Patient}$, we

[3] In logical formulas or lambda-expressions, people often omit the type labels of events and entities: for example, (2) would just be written as $\exists e.\ \mathit{kiss}(e)\ \&\ \mathit{agent}(e, j)\ \&\ \mathit{patient}(e, m)\ \&\ \mathit{passionate}(e)$, since traditionally there are only one type of events and one type of entities; we shall put in the type labels explicitly. Another note on notations is: **e** and **t** in boldface stand for the type of entities and the type of truth values, respectively, as in MG, while $e$ and $t$ not in boldface stand for different things (for example, $e$ would usually be used as a variable of an event type).

[4] Please note here that, for $\mathit{kiss}(e)$ and $\mathit{passionate}(e)$ to be well-typed, the type of event $e$ must be the same as the domain of *kiss* and *passionate* – see the next section about subtyping, which allows them to be well-typed.

can define functions $\text{AGENT}_{AP}[a,p]$ and $\text{PATIENT}_{AP}[a,p]$ such that, for any event $e : Evt_{AP}(a,p)$, $\text{AGENT}_{AP}[a,p](e) = a$ and $\text{PATIENT}_{AP}[a,p](e) = p$.[5]

The parameters of dependent event types are usually names of thematic roles such as agents and patients. Formally, the dependent event types are parameterised by objects of types $A_1, ..., A_n$. Event types with $n$ parameters are called $n$-ary event types. In this paper, we shall only consider $n$-ary event types with $n = 0, 1, 2$:

- When $n = 0$, the event type, usually written as $Event$, has no parameters. $Event$ corresponds to the type of all events in the traditional setting.
- When $n = 1$, we only consider $Evt_A(a)$ and $Evt_P(p)$, where $a : Agent$ and $p : Patient$; i.e., these are event types dependent on agents $a$ and those dependent on patients $p$. For example, if John is an agent with interpretation $j$, $Evt_A(j)$ is the type of events whose agents are John.
- When $n = 2$, we only consider $Evt_{AP}(a,p)$ for $a : Agent$ and $p : Patient$, i.e., the event type dependent on agent $a$ and patients $p$. For example, if agent John and patient Mary, $Evt_{AP}(j,m)$ is the type of events whose agents and patients are John and Mary, respectively (cf., the example (3) above).

Introducing dependent event types has several advantages. In this paper, we shall detail one of them, that is, it gives a natural solution to the event quantification problem – see §5. Before doing that, we shall first consider in §3 the subtyping relationship between event types which, among other things, simplifies the semantic interpretations of VPs in the semantics with dependent event types, and then in §4 the formal systems that underly the proposed semantic treatments and their meta-theoretic properties.

## 3 Subtyping between Event Types

Event types have natural subtyping relationships between them. For example, an event whose agent is $a$ and patient is $p$ is an event with agent $a$. In other words, for $a : Agent$ and $p : Patient$, the type $Evt_{AP}(a,p)$ is a subtype of $Evt_A(a)$. If we only consider the event types $Event$, $Evt_A(a)$, $Evt_P(p)$ and $Evt_{AP}(a,p)$ (cf., the last section), they have the following subtypnig relationships:

$$Evt_{AP}(a,p) \leq Evt_A(a) \leq Event$$

$$Evt_{AP}(a,p) \leq Evt_P(p) \leq Event$$

which can be depicted as Figure 1.

Formally, the subtyping relationship obeys the following rule (called subsumption rule):

$$(*) \qquad \frac{a : A \quad A \leq B}{a : B}$$

[5] Formally, we have $\text{AGENT}_{AP}[a,p] = \lambda e{:}Evt_{AP}(a,p).a$, of type $Evt_{AP}(a,p) \to Agent$. Usually we simply write, for example, $\text{AGENT}_{AP}(e)$ for $\text{AGENT}_{AP}[a,p](e)$.

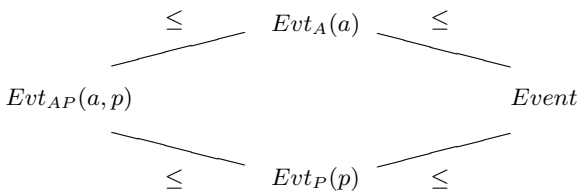


**Fig. 1.** Subtyping between event types with $a : Agent$ and $p : Patient$.

It is also reflexive and transitive. The underlying type theory for formal semantics can be extended by dependent event types together with the subtyping relations. The underlying type theory can either be the simple type theory [5] in the Montagovian semantics or a Modern Type Theory such as UTT [11] in MTT-semantics as considered in, for example, [13]. If the former, extending it with dependent event types results in the formal system $C_e$ with subsumptive subtyping, and if the latter, the resulting theory is UTT[E] with coercive subtyping whose basic coercion relationships in E characterise the subtyping relationships between event types (see §4 for more details).[6]

The incorporation of subtyping between event types is not only natural but plays an essential role in semantic interpretations. This can best be explained by considering how verb phrases are interpreted. In the neo-Davidson's event semantics (with only *Event* as the type of events), a verb phrase is interpreted as a predicate over events, as the following example shows.

(4) $talk : Event \to \mathbf{t}$.

(5) John talked loudly.

(6) $\exists e : Event.\ talk(e)\ \&\ loud(e)\ \&\ agent(e, j)$

With dependent event types such as $Evt_A(j)$, how can we interpret *talk* and (5)? In analogy, the desired semantics of (5) would be (7), where the agent of the event $e$ can be obtained as $\textsc{agent}_A(e) = j$:

(7) $\exists e : Evt_A(j).\ talk(e)\ \&\ loud(e)$

However, if *talk* is of type $Event \to \mathbf{t}$, $talk(e)$ would be ill-typed since $e$ is of type $Evt_A(j)$, not of type *Event*. Is (7) well-typed? The answer is, if we do not have subtyping, it is not. But, if we have subtyping as described above, it is! To elaborate, because $e : Evt_A(j) \leq Event$, $talk(e)$ is well-typed by the subsumption rule $(*)$. Similarly, we have $loud : Event \to \mathbf{t}$ and, therefore, $loud(e)$ is well-typed for $e : Evt_A(j) \leq Event$ as well.

[6] It may be worth mentioning that, in the setting of MTT-semantics, coercive subtyping [12, 14] is used and, for uniformity, we may adopt coercive subtyping rather than subsumptive subtyping, although in general subsumptive subtyping is simpler.

To summarise, the subtyping relations have greatly simplified the event semantics in the presence of refined dependent event types.

*Remark 1.* The subtyping relations also facilitate a natural relationship between the functions such as $\text{AGENT}_{AP}$ and $\text{AGENT}_{A}$ (see §2 and Footnote 5). For example, because of the subtyping relations as depicted in Fig 1, for $e : Evt_{AP}(a, p) \leq Evt_A(a)$, we have, by definition: $\text{AGENT}_{AP}[a, p](e) = \text{AGENT}_A[a](e) = a$.

## 4 The Underlying Systems $C_e$ and UTT[E]

In this section, we describe the formal systems $C_e$ and UTT[E]: $C_e$ extends the simple type theory [5] and UTT[E] extends the modern type theory UTT [11], both with dependent event types and their subtyping relationships as informally described in §2 and §3.[7] $C_e$ is the underlying type theory when we consider formal semantics in the traditional Montagovian setting (as familiar by most of the linguistic semanticists) and UTT[E] when we consider formal semantics in a modern type theory (see, for example, [13, 4]). We also outline the construction of an embedding of $C_e$ into UTT[E] that shows that, like UTT[E], $C_e$ has nice meta-theoretic properties such as normalisation and logical consistency.[8]

### 4.1 The Types System $C_e$

We shall first explain what a context is and what a judgement is in the system $C_e$, and then describe the rules of $C_e$.

*Contexts.* A context is a sequence of entries either of the form $x : A$ or of the form $P\ true$. Informally, the former assumes that the variable $x$ be of type $A$ and the latter that the proposition $P$ be true. Only valid contexts are legal and context validity is governed by the following rules:

$$\frac{}{\langle\rangle\ valid} \qquad \frac{\Gamma \vdash A\ type \quad x \notin FV(\Gamma)}{\Gamma,\ x{:}A\ valid} \qquad \frac{\Gamma \vdash P : \mathbf{t}}{\Gamma,\ P\ true\ valid}$$

where $\langle\rangle$ is the empty sequence and $FV(\Gamma)$ is the set of free variables in $\Gamma$ defined as: (1) $FV(\langle\rangle) = \emptyset$; (2) $FV(\Gamma, x{:}A) = FV(\Gamma) \cup \{x\}$; (3) $FV(\Gamma, P\ true) = FV(\Gamma)$.

*Judgements.* Judgements are sentences in $C_e$, whose correctness are governed by the inference rules below. In $C_e$, there are five forms of judgements:

- $\Gamma\ valid$, which means that $\Gamma$ is a valid context (the rules of deriving context validity are given above).

[7] A notational remark: in $C_e$, $C$ stands for 'Church' and $e$ for 'event'. The notation UTT[E] comes from the work of coercive subtyping (see, for example, [14]) where T[C] denotes type theory T extended by coercive subtyping whose basic subtyping are given as the set C of subtyping judgements.

[8] This section is rather formal and, for a reader less interested in formal matters, its details might be safely skipped if one wishes so.

$$\frac{\Gamma \ valid}{\Gamma \vdash \mathbf{e} \ type} \qquad \frac{\Gamma \ valid}{\Gamma \vdash \mathbf{t} \ type} \qquad \frac{\Gamma,\ x{:}A,\ \Gamma' \ valid}{\Gamma,\ x{:}A,\ \Gamma' \vdash x : A} \qquad \frac{\Gamma,\ P \ true,\ \Gamma' \ valid}{\Gamma,\ P \ true,\ \Gamma' \vdash P \ true}$$

$$\frac{\Gamma \vdash A \ type \quad \Gamma \vdash B \ type}{\Gamma \vdash A \to B \ type} \qquad \frac{\Gamma, x{:}A \vdash b : B \quad x \notin FV(B)}{\Gamma \vdash \lambda x{:}A.b : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$$

$$\frac{\Gamma \vdash P : \mathbf{t} \quad \Gamma \vdash Q : \mathbf{t}}{\Gamma \vdash P \supset Q : \mathbf{t}} \qquad \frac{\Gamma,\ P \ true \vdash Q \ true}{\Gamma \vdash P \supset Q \ true} \qquad \frac{\Gamma \vdash P \supset Q \ true \quad \Gamma \vdash P \ true}{\Gamma \vdash Q \ true}$$

$$\frac{\Gamma \vdash A \ type \quad \Gamma, x{:}A \vdash P : \mathbf{t}}{\Gamma \vdash \forall(A, x.P) : \mathbf{t}} \qquad \frac{\Gamma, x{:}A \vdash P \ true}{\Gamma \vdash \forall(A, x.P) \ true} \qquad \frac{\Gamma \vdash \forall(A, x.P[x]) \ true \quad \Gamma \vdash a : A}{\Gamma \vdash P[a] \ true}$$

**Fig. 2.** Rules for Church's STT.

- $\Gamma \vdash A \ type$, which means that $A$ is a type under context $\Gamma$.
- $\Gamma \vdash a : A$, which means that $a$ is an object of type $A$ under context $\Gamma$.
- $\Gamma \vdash P \ true$, which means that $P$ is a true proposition under context $\Gamma$.
- $\Gamma \vdash A \leq B$, which means that $A$ is a subtype of $B$ under context $\Gamma$.

*Inference rules.* The inference rules for $C_e$ consist of:

1. Rules for context validity (the three rules above);
2. Figure 2: the rules for Church's simple type theory including those for (1) the basic types $\mathbf{e}$ and $\mathbf{t}$ of entities and truth values, (2) function types with $\beta$-conversion ($(\lambda x{:}A.b[x])(a) \simeq b[a]$), and (3) logical formulas[9]; and
3. Figure 3: the rules for dependent event types including those for (1) dependent event types and (2) their subtyping relations, and (3) general subtyping rules including subsumption.

Some explanations of the rules are in order:

- In the $\lambda$-rule in Figure 2, we have added a side condition $x \notin FV(B)$, i.e., $x$ does not occur free in $B$. This is necessary because we have dependent event types like $Evt_A(a)$: for example, we need to forbid to derive $\Gamma \vdash (\lambda x{:}Agent.\lambda e{:}Evt_A(x).e) : Agent \to Evt_A(x) \to Evt_A(x)$ from $\Gamma, x{:}Agent \vdash (\lambda e{:}Evt_A(x).e) : Evt_A(x) \to Evt_A(x)$, where in the former judgement, $x$ in $Agent \to Evt_A(x) \to Evt_A(x)$ would be a free variable that has not been declared in $\Gamma$. Note that, in Church's formulation [5], the side condition is not needed because, there, there are no dependent types (and $x$ does not occur free in $B$ for sure).
- In the rules in Figure 3, since all of the judgements have the same contexts, we have omitted the contexts. For example, the first rule in its third row should have been, if written in full:

$$\frac{\Gamma \vdash a : Agent \quad \Gamma \vdash p : Patient}{\Gamma \vdash Evt_{AP}(a, p) \leq Evt_A(a)}$$

[9] We only consider the intuitionistic $\supset$ and $\forall$ here, omitting other operators including, in particular, those about, e.g. negation/classical logic in [5]. Also, we shall not assume extensionality.

$$\frac{}{Agent\ type} \quad \frac{}{Patient\ type}$$

$$\frac{}{Event\ type} \quad \frac{a : Agent}{Evt_A(a)\ type} \quad \frac{p : Patient}{Evt_P(p)\ type} \quad \frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p)\ type}$$

$$\frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p) \leq Evt_A(a)} \quad \frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p) \leq Evt_P(p)}$$

$$\frac{a : Agent}{Evt_A(a) \leq Event} \quad \frac{p : Patient}{Evt_P(p) \leq Event}$$

$$\frac{A\ type}{A \leq A} \quad \frac{A \leq B \quad B \leq C}{A \leq C} \quad \frac{A' \leq A \quad B \leq B'}{A \to B \leq A' \to B'}$$

$$\frac{A \simeq B}{A \leq B} \quad \frac{a : A \quad A \leq B}{a : B}$$

**Fig. 3.** Rules for dependent event types.

### 4.2 The Type System UTT[E]

The type theory UTT (Chapter 9 of [11]) is a dependent type theory with inductive types, type universes and higher-order logic. UTT is a typical Modern Type Theory (MTT) as employed in MTT-semantics [13, 4] (actually, it is the MTT the first author and colleagues have employed in developing MTT-semantics). Its meta-theory was studied in the PhD thesis by Healf Goguen [8]. Coercive subtyping [12, 14] has been developed by the authors and colleagues for modern type theories such as Martin-Löf's type theory and UTT.

Besides the type constructors in UTT as described in [11], UTT[E] has the following constant types and constant type families for dependent event types:

- $Entity : Type$
- $Agent,\ Patient : Type.$
- $Event : Type,$
  $Evt_A : (Agent)Type,$
  $Evt_P : (Patient)Type,$ and
  $Evt_{AP} : (Agent)(Patient)Type.$

The coercive subtyping relations in UTT[E] are given by subtyping judgements in E: they specify the subtyping relationships between dependent event types by means of the following parameterised constant coercions $c_i$ $(i = 1, ..., 4)$ in E, where $a : Agent$ and $p : Patient$:

$$Evt_{AP}(a,p) \leq_{c_1[a,p]} Evt_A(a), \quad Evt_{AP}(a,p) \leq_{c_2[a,p]} Evt_P(p),$$

$$Evt_A(a) \leq_{c_3[a]} Event, \quad Evt_P(p) \leq_{c_4[p]} Event,$$

The coercions also satisfy the coherence condition $c_3[a] \circ c_1[a,p] = c_4[p] \circ c_2[a,p]$.

Based on the study in [14, 21], it is straightforward to show that UTT[E] is a well-behaved extension of UTT and hence preserves its nice meta-theoretic

properties, including Church-Rosser, subject reduction, strong normalisation, and logical consistency.

*Remark 2.* As mentioned above, UTT[E] underlies the development of MTT-semantics by the first author and colleagues [13, 4]. In the recent trend of using rich type theories in formal semantics (see, for example, some of the papers in [3]), the development of MTT-semantics provides a full-blown alternative to the traditional Montague semantics with many advantages and has its further potentials to be developed in the future. It is worth remarking that UTT[E] underlies the event semantics in dependent type theories (or MTT-semantics with events) which contain, in particular, dependent event types.

### 4.3 Embedding of $\mathcal{C}_e$ into UTT[E]

In this subsection, we show that $\mathcal{C}_e$ can be faithfully embedded into UTT[E] and hence has nice meta-theoretic properties. The embedding of $\mathcal{C}_e$ into UTT[E] is defined as follows and it is faithful as the theorem below shows.

**Definition 1 (embedding)** *The embedding $[\![\_]\!]$ from $\mathcal{C}_e$ to UTT[E] is inductively defined as follows:*[10]

1. *Constant types and dependent event types:*
   - $[\![\boldsymbol{e}]\!]_\Gamma = Entity$.
   - $[\![\boldsymbol{t}]\!]_\Gamma = Prop$.

   *For the other constant types and dependent event types, they are mapped to the 'same' types in UTT[E], since we have overloaded their names. For example,*
   - $[\![Agent]\!] = Agent$
   - $[\![Evt_A(a)]\!] = Evt_A([\![a]\!])$
2. *Non-constant terms:*
   - $[\![x]\!]_\Gamma = x$
   - $[\![A \to B]\!]_\Gamma = [\![A]\!]_\Gamma \to [\![B]\!]_\Gamma$
   - $[\![\lambda x{:}A.b]\!]_\Gamma = \lambda([\![A]\!]_\Gamma, T, [x{:}[\![A]\!]_\Gamma]\,[\![b]\!]_{\Gamma,x:A})$, *if* $[\![\Gamma, x{:}A]\!] \vdash [\![b]\!]_{\Gamma,x:A} : T$
   - $[\![f(a)]\!]_\Gamma = app(S, T, [\![f]\!]_\Gamma, [\![a]\!]_\Gamma)$, *if* $[\![\Gamma]\!] \vdash [\![f]\!]_\Gamma : S \to T$ *and* $[\![\Gamma]\!] \vdash [\![a]\!]_\Gamma : S_0$, *where* $[\![\Gamma]\!] \vdash S_0 \leq S$
   - $[\![P \supset Q]\!]_\Gamma = [\![P]\!]_\Gamma \supset [\![Q]\!]_\Gamma$
   - $[\![\forall(A, x.P)]\!]_\Gamma = \forall([\![A]\!]_\Gamma, [x{:}[\![A]\!]_\Gamma].\,[\![P]\!]_{\Gamma,x:A})$
3. *Contexts:*
   - $[\![\langle\rangle]\!] = \langle\rangle$ *(the empty context in UTT[E])*
   - $[\![\Gamma,\ x : A]\!] = [\![\Gamma]\!],\ x : [\![A]\!]_\Gamma$
   - $[\![\Gamma,\ P\ true]\!] = [\![\Gamma]\!],\ x : \mathbf{Prf}([\![P]\!]_\Gamma)$, *where* $x$ *does not occur free in* $[\![\Gamma]\!]$.

[10] Formally, this is a partial function – it is only defined when certain conditions hold. The embedding theorem shows that the embedding is total for well-typed terms. Also, a notional note: we shall use $S$ and $T$ to stand for types in UTT[E] where function types are special cases of $\Pi$-types: for any types $S$ and $T$, $S \to T = \Pi(S, [\_{:}S]T)$.

The following theorem shows that the embedding is well-defined and faithful (in the sense of the theorem) and hence $C_e$ has nice meta-theoretic properties (the corollary). Its proof is based on the embedding of Church's simple type theory into the calculus of constructions [10]. We omit the discussion of technical details, for otherwise we would have to detail the syntax and rules of UTT and coercive subtyping [11, 14], except remarking that a key reason that the proof goes through is because the coercions to model subtyping for dependent event types are constants and coherent (see §4.2) and hence models subsumptive subtyping in $\mathcal{C}_e$ faithfully.

**Theorem 2 (faithfulness)** *The embedding in Definition 1 is defined for every well-typed term in $\mathcal{C}_e$ and, furthermore, we have:*

1. *If $\Gamma$ valid in $\mathcal{C}_e$, then $[\![\Gamma]\!]$ valid in UTT[E].*
2. *If $\Gamma \vdash A$ type in $\mathcal{C}_e$, then $[\![\Gamma]\!] \vdash [\![A]\!] : Type$ in UTT[E].*
3. *If $\Gamma \vdash a : A$ in $\mathcal{C}_e$, then in UTT[E], $[\![\Gamma]\!] \vdash [\![a]\!] : T$ for some $T$ such that $[\![\Gamma]\!] \vdash T \leq_d [\![A]\!]$ for some $d$.*
4. *If $\Gamma \vdash P$ true in $\mathcal{C}_e$, then $[\![\Gamma]\!] \vdash p : \mathbf{Prf}([\![P]\!])$ for some $p$ in UTT[E].*
5. *If $\Gamma \vdash A \leq B$ in $\mathcal{C}_e$, then $[\![\Gamma]\!] \vdash [\![A]\!] \leq_c [\![B]\!]$ for some unique $c$ in UTT[E].*

**Corollary 3** *$\mathcal{C}_e$ inherits nice meta-theoretic properties from UTT[E], including strong normalisation and logical consistency.*

*Remark 3.* Instead of the embedding method we have described here, one may consider a more direct approach to metatheory of $\mathcal{C}_e$ by directly showing that it has nice properties such as Church-Rosser and strong normalisation (as suggested by an anonymous reviewer). However, we think the above is simpler, which is of course a subjective view, and also demonstrates a generic approach to such meta-theoretic studies.

## 5 Event Quantification Problem

It is known that, when considering (neo-)Davidsonian event semantics where existential quantifiers for event variables are introduced, there is a problem in dealing with the scopes of the quantifiers when other quantificational phrases are involved. It has been argued that there is some incompatibility between event semantics and the traditional compositional semantics [2, 20]. De Groote and Winter [7] have called this as the *event quantification problem* (EQP for short).

Consider the following sentence (8) which, under the traditional event semantics with $bark : Event \to \mathbf{t}$, could have two possible interpretations (9) and (10), where (10) is incorrect.

(8) No dog barks.

(9) $\neg\exists x : \mathbf{e}.\ dog(x)\ \&\ \exists e{:}Event.\ bark(e)\ \&\ agent(e, x)$

(10) (#) $\exists e : Event.\ \neg\exists x : \mathbf{e}.\ dog(x)\ \&\ bark(e)\ \&\ agent(e, x)$

Formally, the incorrect interpretation is acceptable just as the correct one: (10) is a legal formula. In order to avoid such incorrect interpretations as (10), people have made several proposals (see, for example, [2, 20]) which involve, for instance, consideration of quantification not over events but over sets of events [2], or some informal (and somewhat *ad hoc*) principles whose adherence would disallow the incorrect interpretations (see, for example, the related Scope Domain Principle proposed by Landman [9]).

We shall study this with dependent event types as informally studied in §2 and §3, both in the Montagovian setting (i.e., in $C_e$ as described in §4.1) and in the MTT-semantics (i.e., in UTT[E] as described in §4.2). It is shown that, with dependent event types, the incorrect semantics are blocked as illegal since they are ill-typed.

### 5.1 EQP in Montague Semantics with Dependent Event Types

In the Montagovian setting with dependent event types (formally, $C_e$ in §4.1), this problem is solved naturally and *formally* – the incorrect semantic interpretations are excluded because they are ill-typed (in the empty context, where semantic interpretations of whole sentences like (8) are considered).

For example, (8) will be interpreted as (11), while the 'incorrect' interpretation (12) is not available (the formula (12) is ill-typed because $x$ in $Evt_A(x)$, outside the scope of second/bound $x$ (although intuitively it refers to it), is a free variable without being declared.)

(11) $\neg\exists x : \mathbf{e}.\ (dog(x)\ \&\ \exists e : Evt_A(x).\ bark(e))$

(12) $(\#)\ \exists e : Evt_A(x).\ \neg\exists x : \mathbf{e}.\ dog(x)\ \&\ bark(e)$

This offers a natural solution to the event quantification problem. Compared with existing solutions with informal ad hoc principles such as those mentioned above, our solution comes naturally as a 'side effect' of introducing dependent event types: it is formally disciplined and natural.

### 5.2 EQP in MTT-semantics with Dependent Event Types

In this paper, we have focussed on extending the traditional Montague semantics with dependent event types (formally, $\mathcal{C}_e$), since the simple type theory is what the most semanticists are familiar with. One can also extend the MTT-semantics [13, 4] with dependent event types (formally, UTT[E], if we use UTT for MTT-semantics) and hence consider such refined event semantics in the setting of MTT-semantics. Here, we give an example to show how this is done.

Still consider the sentence (8): No dog barks. In the MTT-semantics, where CNs are interpreted as types (rather than predicates), the verb `bark` is given a dependent type as its semantics:

(13) $bark : \Pi x{:}Dog.\ Evt_A(x) \to Prop$

It is also the case that the correct semantics (14) for (8) is legal (well-typed), while the incorrect one (15) is not:

(14) $\neg\exists x : Dog.\ \exists e : Evt_A(x).\ bark(x, e)$

(15) (#) $\exists e : Evt_A(x).\ \neg\exists x : Dog.\ bark(x, e)$

Note that (15) is ill-typed for two reasons now: the first $x$ is a variable not assumed anywhere and the term $bark(x, e)$ is ill-typed as well.

Employing dependent event types in the Montagovian semantics (i.e., in $C_e$ as described in §4.1), would still leave a small possibility of some formally legal but incorrect semantics. For instance, one might consider the following semantics for (8):

(16) (#) $\exists e : Event.\ \neg\exists x : \mathbf{e}.\ dog(x)\ \&\ bark(e)$

Note that, although (16) is incorrect, it is still well-typed because $e$ is just an event, not an event with $x$ as agent.[11] This, however, would not happen in the MTT-semantic setting where the type of the verb bark is the dependent type (13) and the following semantic sentence is ill-typed:

(17) (#) $\exists e : Event.\ \neg\exists x : Dog.\ bark(x, e)$,

because $bark(x, e)$ is not well-typed (it requires $e$ to be of type $Evt_A(x)$, not just of type $Event$).

## 6 Conclusion

In this paper, we have introduced dependent event types for formal semantics. Subtyping is shown to play an essential role in this setting. We have also considered how dependent event types naturally solve the event quantification problem in combining event semantics with the traditional compositional semantics.

The notion of event types as studied in this paper is *intensional*, rather than extensional. For instance, when considering inverse verb pairs such as `buy` and `sell`, one may think that the events in (18) and (19) are the same [19].

(18) John bought the book from Mary.

(19) Mary sold the book to John.

If one considers this from the angle of extensionality/intensionality, the buying event and the selling event in the above situation are extensionally the same, but intensionally different. More generally, this is related to how to understand the sameness of events in the setting with dependent event types. Work need be done to study event structures and relevant inference patterns.

Another interesting research topic is to study whether all thematic roles should be considered as parameters of dependent event types. Unlike agents and

[11] Of course, one can argue that this is not intended since the agent is known, but formally, nothing prevents one from doing it.

patients, some thematic roles considered in the literature may not be suitable to play the role of indexing dependent event types. In such cases, we would still propose that they should be formalised by means of logical predicates/relations. In the other direction, event types may depend on other entities other than thematic roles and further studies are called for to understand this better.

**Acknowledgement**. Thanks go to Stergios Chatzikyriakidis, David Corfield, Koji Mineshima and Christian Retoré for helpful comments on this work.

## References

1. N. Asher and Z. Luo. Formalisation of coercions in lexical semantics. *Sinn und Bedeutung 17, Paris*, 2012.
2. L. Champollion. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, 38:31–66, 2015.
3. S. Chatzikyriakidis and Z. Luo, editors. *Modern Perspectives in Type-Theoretical Semantics*. Springer, 2017.
4. S. Chatzikyriakidis and Z. Luo. *Formal Semantics in Modern Type Theories*. ISTE/Wiley, 2018. (to appear).
5. A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(1), 1940.
6. D. Davidson. The logical form of action sentences. *In: S. Rothstein (ed.). The Logic of Decision and Action. University of Pittsburgh Press*, 1967.
7. P. de Groote and Y. Winter. A type-logical account of quantification in event semantics. *Logic and Engineering of Natural Language Semantics 11*, 2014.
8. H. Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, 1994.
9. F. Landman. Plurality. In S. Lappin, editor, *The Handbook of Contemporary Semantic Theory*, 1996.
10. Z. Luo. A problem of adequacy: conservativity of calculus of constructions over higher-order logic. Technical report, LFCS report series ECS-LFCS-90-121, Department of Computer Science, University of Edinburgh, 1990.
11. Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.
12. Z. Luo. Coercive subtyping in type theory. *CSL'96, LNCS'1258*, 1997.
13. Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.
14. Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 223:18–42, 2012.
15. P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
16. R. Montague. *Formal Philosophy*. Yale University Press, 1974. Collected papers edited by R. Thomason.
17. T. Parsons. *Events in the Semantics of English*. MIT Press, 1990.
18. A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
19. A. Williams. *Arguments in Syntax and Semantics*. Cambridge, 2015.
20. Y. Winter and J. Zwarts. Event semantics and abstract categorial grammar. *Proc. of Mathematics of Language 12, LNCS 6878*, 2011.
21. T. Xue. *Theory and Implementation of Coercive Subtyping*. PhD thesis, Royal Holloway, University of London, 2013.