Dependent Event Types[⋆]

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

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

November 30, 2016

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. The underlying formal system C_e for dependent event types, that extends the simple type theory as employed in Montague semantics, is described and proven to have nice meta-theoretic properties such as normalisation and logical consistency. It is also 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 formal semantics in modern type theories.

1 Introduction

The event semantics, whose study was initiated by Davidson [4] and further studied in its neo-Davidsonian turn (see [14] 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 [12] and UTT [8], provide a useful tool in formalising event types and provide a nice treatment of the event semantics.

An event type may depend on thematic roles such as agents and patients of the events in the type. For example, we can consider the type $Evt_{AP}(a,p)$ of events whose agent and patient are a and p, respectively. It is shown that such dependent event types give a natural solution to the incompatibility problem

⋆ This is the online paper to accompany the LACL 2016 abstract [10]. The main difference with the earlier online paper on September 23, with the same title, is that we have now added §4 and §5.2.

⋆⋆ This work is partially supported by EU COST Action CA15123 and CAS/SAFEA International Partnership Program.

in combining event semantics with the traditional Montague semantics [2, 17] (sometimes called the event quantification problem [5]): the correct semantics are accepted while the incorrect ones are excluded by typing because they would be ill-typed and hence illegal.

We shall investigate subtyping relations between event types which include dependent types such as $Evt(a, p)$ 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.

Dependent event types 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 short paper, we shall study event types dependent on thematic roles in a general setting in formal semantics with events both in the traditional Montague semantics [13] and in formal semantics in modern type theories [15, 9]. In §2, we shall describe the basics of dependent event types, 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 dependent event types. §5 shows that a natural solution to the event quantification problem can be found in the setting with dependent event types, giving an example of advantages over the traditional setting. In the concluding section, we shall briefly discuss the future work on dependent event types.

2 Dependent Event Types

In the Davidsonian event semantics in the traditional Montagovian setting [4, 14], there is only one type $Event$ of all events. For example, the sentence (1) is interpreted as (2):

(1) John kissed Mary passionately.

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

where in (2), $Event$ is the type of all events, $kiss$, $passionate : Event \rightarrow \mathbf{t}$ are predicates over events, and $agent$, $patient : Event \rightarrow \mathbf{e} \rightarrow \mathbf{t}$ are relations between events and entities.[3] Please note that, in the above neo-Davidson's

[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 $\exists e.\ kiss(e)\ \&\ agent(e, j)\ \&\ patient(e, m)\ \&\ passionate(e)$, 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: $\mathbf{e}$ and $\mathbf{t}$ in boldface stand for the type of entities and the type of true 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).

semantics (2), adverbial modifications and thematic role relations are all propositional conjuncts in parallel with the verb description, an advantageous respect 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 $a : Agent$ and $p : Patient$, the dependent type

$$Evt_{AP}(a, p)$$

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) $\exists e : Evt_{AP}(j, m).\ kiss(e)\ \&\ passionate(e)$

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 $a : Agent$ and $p : Patient$, we can define functions $\textsc{agent}_{AP}[a, p]$ and $\textsc{patient}_{AP}[a, p]$ such that, for any event $e : Evt_{AP}(a, p)$, $\textsc{agent}_{AP}[a, p](e) = a$ and $\textsc{patient}_{AP}[a, p](e) = p$.[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 consider the subtyping relationship between event types which, among other things, simplifies the semantic interpretations of VPs in the semantics with dependent event types.

[4] Please note here that, for $kiss(e)$ and $passionate(e)$ 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.

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

$$\begin{tikzcd}[arrows=dash]
 & Evt_A(a) \arrow[dr, "\leq"] & \\
Evt_{AP}(a,p) \arrow[ur, "\leq"] \arrow[dr, "\leq"'] & & Event \\
 & Evt_P(p) \arrow[ur, "\leq"'] &
\end{tikzcd}$$

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

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):

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

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 governed by the subsumption rule.[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) $talk : Event \rightarrow \mathbf{t}$.

(5) John talked loudly.

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

[6] See §4 for the underlying formal system C_e, which is the extension of Church's simple type theory [3] with dependent event types.

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 $Event \rightarrow \mathbf{t}$, $talk(e)$ 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 $loud : Event \rightarrow \mathbf{t}$ and, therefore, $loud(e)$ is well-typed for $e : Evt_A(j) \leq Event$ as well.

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 $\textsc{agent}_{AP}$ and $\textsc{agent}_A$ (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: $\textsc{agent}_{AP}[a, p]e) = \textsc{agent}_A[a](e) = a$.

4 The Underlying System C_e and Its Embedding

In this section, we describe the underlying formal system C_e for dependent event types, that extends the simple type theory as employed in Montague semantics, and proves that C_e has nice meta-theoretic properties such as normalisation and logical consistency, by embedding C_e into a modern type theory (see below).[7]

C_e is the extension of Church's simple type theory [3], as used in the Montague semantics, with dependent event types and the subtyping relations between them, as informally described in §2 and §3. C_e can be faithfully embedded into UTT[C], i.e., the type theory UTT [8] extended with coercive subtyping in C [11], where C contains the subtyping judgements that correspond to the subtyping relations between dependent event types as described in §3. Since UTT[C] has nice meta-theoretic properties such as normalisation and logical consistency, so does C_e.

In §4.1, we shall give a formal presentation of C_e and then, in §4.2, prove that it can be faithfully embedded into UTT[C] and hence has nice properties.

4.1 The Underlying System C_e

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

[7] This section is rather formal and its details might be safely skipped if wished.

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

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

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

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

Fig. 2. Rules for Church's STT.

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 Γ.[8]

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

- Γ *valid*, which informally means that Γ is a valid context (the rules of deriving context validity) are given above).
- $\Gamma \vdash A$ *type*, which informally means that A is a type under context Γ.
- $\Gamma \vdash a : A$, which informally means that a is an object of type A under context Γ.
- $\Gamma \vdash P$ *true*, which informally means that P is a true proposition under context Γ.
- $\Gamma \vdash A \leq B$, which informally means that A is a subtype of B under context Γ.

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 **e** and **t** of entities and formulas, (2) function types with β-conversion ($(\lambda x{:}A.b[x])(a) \simeq b[a]$), and (3) logical formulas[9]; and

[8] Formally, $FV(\Gamma)$ is inductively defined as follows: (1) $FV(\langle\rangle) = \emptyset$; (2) $FV(\Gamma,\ x{:}A) = FV(\Gamma) \cup \{x\}$; (3) $FV(\Gamma,\ P\ true) = FV(\Gamma)$.

[9] We only consider the intuitionistic $\supset$ and $\forall$ here, omitting other operators and, in particular, those about, e.g. negation/classical logic in the original version of [3]. 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.

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 λ-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 $\Gamma \to \lambda x{:}Agent.e : Agent \to Evt_A(x)$ from $\Gamma, x{:}Agent \vdash e : Evt_A(x)$, where in the former judgement, x would not have been declared in Γ.
– 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:

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

4.2 Embedding of C_e into UTT[C]

In this subsection, we show that $\mathcal{C}_e$ can be faithfully embedded into UTT[C] and hence have nice meta-theoretic properties. The type theory UTT [8] is a dependent type theory with inductive types, type universes and higher-order logic.[10] Its meta-theory was studied in the PhD thesis by Healf Goguen [6] where, in particular, it is shown that UTT is strongly normalising and hence logically consistent. Coercive subtyping has been developed by the authors and

[10] If one has had contacts with formal semantics in modern type theories (MTT-semantics, see, for example, [9]), UTT is a typical MTT – actually, it is the MTT the first author and colleagues have employed in developing MTT-semantics.

colleagues for dependent type theories such as Martin-Löf's type theory and UTT [11]. The fact that $\mathcal{C}_e$ can be faithfully embedded into UTT[C], where C contains the subtyping judgements that correspond to the subtyping relations between dependent event types, shows that $\mathcal{C}_e$ has nice meta-theoretic properties as well; in particular, $\mathcal{C}_e$ is also strongly normalising and logically consistent.

We shall not describe UTT here and interested readers can study the given references (for example, about MTT-semantics). Before the embedding theorem, we shall describe the coercive subtyping relations in UTT[C], as specified in C.

Subtyping in UTT[C]. In UTT[C], we have 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 dependent event types have subtyping relationships specified by the following parameterised constant coercions c_i $(i = 1, ..., 4)$ in C, where $a : Agent$ and $p : Patient$:

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

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

which also satisfy the following coherence condition: $c_3[a] \circ c_1[a,p] = c_4[p] \circ c_2[a,p]$.

The embedding of $\mathcal{C}_e$ into UTT[C] is defined as follows and it is a faithful embedding as the theorem below shows.

Definition 1 (embedding) *The embedding $[\![\_]\!]$ from $\mathcal{C}_e$ to UTT[C] is inductively defined as follows:*[11]

- $[\![x]\!]_\Gamma = x.$
- $[\![\boldsymbol{e}]\!]_\Gamma = Entity.$
- $[\![\boldsymbol{t}]\!]_\Gamma = Prop.$
- $[\![A \to B]\!]_\Gamma = [\![A]\!]_\Gamma \to [\![B]\!]_\Gamma.$
- $[\![\lambda x{:}A.b]\!]_\Gamma = \lambda([\![A]\!]_\Gamma, T, [x{:}\,[\![A]\!]_\Gamma]\ [\![b]\!]_{\Gamma,x:A})$, *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* $[\![\Gamma]\!] \vdash S_0 \leq S.$

[11] 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[C] where function types are special cases of Π-types: for any types S and T, $S \to T = \Pi(S, [\_{:}S]T)$.

– $[\![P \supset Q]\!]_\Gamma = [\![P]\!]_\Gamma \supset [\![Q]\!]_\Gamma$
– $[\![\forall(A, x.P)]\!]_\Gamma = \forall([\![A]\!]_\Gamma, [x{:}\, [\![A]\!]_\Gamma].\, [\![P]\!]_{\Gamma,x:A})$

Furthermore, dependent event types are mapped to the 'same' types in UTT[C] (we have overloaded their names). The embedding is extended to contexts in a straightforward way.

The following theorem shows that the embedding is well-defined and faithful (in the sense of the theorem). Its proof is based on the embedding of Church's simple type theory into the calculus of constructions [7], but has proved to be very intriguing, partly because that, embedding $\mathcal{C}_e$ into UTT[C], we need to consider how to emulate subsumptive subtyping by means of coercive subtyping and how to use fully typed terms of the forms such as $\lambda(S, T, f)$ and $app(A, B, f, a)$ to emulate ordinary λ-terms of the forms $\lambda x{:}A.b$ and $f(a)$. Both of these are interesting in themselves, but the technical details are out of the scope of the current paper.

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 Γ valid in $\mathcal{C}_e$, then $[\![\Gamma]\!]$ valid in UTT[C].*
2. *If $\Gamma \vdash A$ type in $\mathcal{C}_e$, then $[\![\Gamma]\!] \vdash [\![A]\!] : Type$ in UTT[C].*
3. *If $\Gamma \vdash a : A$ in $\mathcal{C}_e$, then in UTT[C], $[\![\Gamma]\!] \vdash [\![a]\!] : T$ 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 $[\![\Gamma]\!] \vdash p : [\![P]\!]$ for some p in UTT[C].*
5. *If $\Gamma \vdash A \leq B$ in $\mathcal{C}_e$, then $[\![\Gamma]\!] \vdash [\![A]\!] \leq_c [\![B]\!]$ for some unique c in UTT[C].*

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

5 Event Quantification Problem

It has been argued that there is some incompatibility between (neo-)Davidsonian event semantics and the traditional compositional semantics [2, 17]. De Groote and Winter [5] have dubbed this as the *event quantification problem* (EQP for short). Consider the following sentence (8) which, under the traditional event semantics, 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) (#) $\exists e : Event.\ \neg\exists x : \mathbf{e}.\ dog(x)\ \&\ bark(e)\ \&\ agent(e, x)$

where $bark : Event \to \mathbf{t}$.

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, 17]) 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.

We shall study this with dependent event types both in the Montagovian setting (i.e., the system informally described in §2 and §3 and formally in C_e as described in §4.1) and in the MTT-semantics (formally in UTT[C] as described in §4.2). It is shown that, with dependent event types, the incorrect semantics is 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), this problem is solved naturally and *formally* – the incorrect semantic interpretations are excluded because they 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 so far, we have only considered extending the traditional Montague semantics with dependent event types (and hence, formally, $\mathcal{C}_e$ – the extension of Church's STT with dependent event types). One can also extend the MTT-semantics [9] with dependent event types 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) $bark : \Pi x{:}Dog.\ Evt_A(x) \rightarrow Prop$

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.

Remark 2. Employing dependent event types in the Montagovian semantics (i.e., in C_e as described in §4), would still leave a small possibility of some formally legal but incorrect semantics.[12] 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.[13] 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 event [16].

(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. Work need be done to study the such event structures and relevant inference patterns. More generally, this is related to how to understand the sameness of events in the setting with dependent event types.

Another interesting research topic is to study whether general thematic roles should be considered as parameters of dependent event types. Unlike agents and 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. However, more careful studies are called for.

Acknowledgement. Thanks go to David Corfield who has spotted an error in a previous version and given useful suggestions, and to Christian Retoré for helpful comments on this work.

[12] Thanks for an anonymous LACL referee for a useful comment on this.

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

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. A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(1), 1940.
4. D. Davidson. The logical form of action sentences. *In: S. Rothstein (ed.). The Logic of Decision and Action. University of Pittsburgh Press*, 1967.
5. P. de Groote and Y. Winter. A type-logical account of quantification in event semantics. *Logic and Engineering of Natural Language Semantics 11*, 2014.
6. H. Goguen. *A Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, 1994.
7. 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.
8. Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science.* Oxford University Press, 1994.
9. Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.
10. Z. Luo and S. Soloviev. Dependent event types (abstract). *Logical Aspects of Computational Linguistics (LACL 2016)*, 2016.
11. Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 223:18–42, 2012.
12. P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.
13. R. Montague. *Formal Philosophy.* Yale University Press, 1974. Collected papers edited by R. Thomason.
14. T. Parsons. *Events in the Semantics of English.* MIT Press, 1990.
15. A. Ranta. *Type-Theoretical Grammar.* Oxford University Press, 1994.
16. A. Williams. *Arguments in Syntax and Semantics.* Cambridge, 2015.
17. Y. Winter and J. Zwarts. Event semantics and abstract categorial grammar. *Proc. of Mathematics of Language 12, LNCS 6878*, 2011.