Factivity and presupposition in Dependent Type Semantics

Ribeka Tanaka[1], *Koji Mineshima*[1,2], *and Daisuke Bekki*[1,2]
[1] Ochanomizu University
[2] CREST, Japan Science and Technology Agency

ABSTRACT

Dependent type theory has been applied to natural language semantics to provide a formally precise and computationally adequate account of dynamic aspects of meaning. One of the frameworks of natural language semantics based on dependent type theory is Dependent Type Semantics (DTS), which focuses on the compositional interpretations of anaphoric expressions. In this paper, we extend the framework of DTS with a mechanism to handle logical entailment and presupposition associated with factive verbs such as *know*. Using the notion of proof objects as first-class objects, we provide a compositional account of presuppositional inferences triggered by factive verbs. The proposal also gives a formal reconstruction of the type-distinction between propositions and facts, and thereby accounts for the lexical semantic differences between factive and non-factive verbs in a type-theoretical setting.

Keywords: dependent type, anaphora, presupposition, proof object, factive verb

1 INTRODUCTION

Dependent Type Semantics (DTS, Bekki 2014) is a framework of natural language semantics based on dependent type theory (Martin-Löf 1984; Nordström *et al.* 1990). In contrast to traditional model-theoretic semantics, DTS is a proof-theoretic semantics, where inference relations between sentences are characterized as provability relations between semantic representations. One of the distinctive features of DTS, as compared to other type-theoretical frameworks, is that it is augmented with underspecified terms, so as to provide a unified analysis of inference, anaphora and presupposition from a logical/computational perspective. In contrast to previous work on anaphora in dependent type theory (cf. Ranta 1994), DTS gives a fully compositional account of inferences involving anaphora. It is also extended to the analysis of modal subordination (Tanaka *et al.* 2015).

In this paper, we provide the framework of DTS with a mechanism to handle logical entailment and presupposition associated with factive verbs. We will mostly focus on the epistemic verb *know*. Although there are numerous studies on factive verbs in natural language semantics, they are usually based on model-theoretic approaches; it seems fair to say that there has been little attempt to formalize inferences with factivity from a proof-theoretical perspective. On the other hand, various proof systems for knowledge and belief have been studied in the context of epistemic logic (cf. Meyer and van der Hoek 2004). However, such systems are mainly concerned with knowledge and belief themselves, not with how they are expressed in natural languages, nor with linguistic phenomena such as factivity presuppositions. Our study aims to fill this gap by providing a framework that explains logical entailment and presuppositions with factive verbs in dependent type theory.

2 DEPENDENT TYPE SEMANTICS

This section introduces the framework of DTS and explains how presuppositions are handled in this framework. In Section 2.1, we provide some necessary background on DTS, including the basics of dependent type theory and the analysis of anaphora within this approach. One of the important problems in the application of dependent type theory to natural language semantics is how to represent common nouns using the machinery of dependent types. Section 2.2 is devoted to discussing this problem. We give several reasons for preferring the view that common nouns are represented as *predicates* rather than as *types*. Given this background, Section 2.3 provides a compositional analysis of presupposition in DTS.

2.1 *Dependent type theory*

In dependent type theory, there are two type constructors, Σ and Π, which play a crucial role in forming the semantic representations for natural language sentences. The type constructor Σ is a generalized form of the product type and behaves as an existential quantifier. An object of type $(\Sigma x : A) B(x)$ is a pair (m, n) such that m is of type A and n is of type $B(m)$. Conjunction $A \wedge B$ is a degenerate form of $(\Sigma x : A) B$ if x does not occur free in B. The Σ-types are associated with projection functions π_1 and π_2 that are computed with the rules $\pi_1(m, n) = m$ and $\pi_2(m, n) = n$, respectively. The type constructor Π is a generalized form of the functional type and behaves as a universal quantifier. An object of type $(\Pi x : A) B(x)$ is a function f such that for any object a of type A, $f a$ is an object of type $B(a)$. Implication $A \to B$ is a degenerate form of $(\Pi x : A) B$ if x does not occur free in B. The inference rules for Π-types and Σ-types are shown in the Appendix.[1] Throughout the paper, we will make use of the DTS-notation for Π-types and Σ-types as shown in Figure 1.

	Π-types	Σ-types
Standard notation	$(\Pi x : A) B(x)$	$(\Sigma x : A) B(x)$
Notation in DTS	$(x{:}A) \to B(x)$	$\begin{bmatrix} x : A \\ B(x) \end{bmatrix}$
When $x \notin fv(B)$	$A \to B$	$\begin{bmatrix} A \\ B \end{bmatrix}$

Figure 1:
Notation for Π-types and Σ-types in DTS ($fv(B)$ means the set of free variables in B)

Based on the Curry-Howard correspondence (Howard 1980), a type can be regarded as a proposition and a term can be regarded as a proof. Thus the judgement $a : A$ can be read as “a is a proof of proposition A”, as well as “a is a term of type A”. In this setting, the truth of a proposition A is defined as the existence of a term of type A. The term that serves as a proof of a proposition is called a *proof term* and plays an important role in representing natural language sentences in dependent type theory.

Since the work of Sundholm (1986) and Ranta (1994), dependent type theory has been applied to the analysis of various dynamic discourse phenomena, providing a type-theoretic alternative to model-theoretic frameworks such as Discourse Representation Theory (van der Sandt 1992; Kamp *et al.* 2011), Dynamic Predicate Logic (Groenendijk and Stokhof 1991), and Dynamic Semantics (Heim

[1] For more details, readers can refer to Martin-Löf (1984) and Ranta (1994).

1983). For instance, according to the analysis presented in Sundholm (1986) and Ranta (1994), a semantic representation for the donkey sentence in (1) can be given as (2) in terms of dependent types.

(1) Every farmer who owns a donkey beats it.

(2) $$\left(u : \begin{bmatrix} x : \mathbf{farmer} \\ \begin{bmatrix} y : \mathbf{donkey} \\ \mathbf{own}(x, y) \end{bmatrix} \end{bmatrix} \right) \to \mathbf{beat}(\pi_1 u, \pi_1 \pi_2 u)$$

The sentence (1) as a whole is a universal sentence, which is represented as a Π-type. The restrictor *farmer who owns a donkey* is analyzed as a Σ-type. A term u having this Σ-type would be a tuple $(f, (d, o))$, where f is a term of type **farmer**, d is a term of type **donkey**, and o is a proof-term of the proposition $\mathbf{own}(f, d)$. Recall that π_1 and π_2 are projection functions that take a pair and return the first and the second element, respectively. Thus the terms $\pi_1 u$ and $\pi_1 \pi_2 u$ appearing in the consequent of (2) pick up from u the term f of type **farmer** and the term d of type **donkey**, respectively. In this way, via the proof term u associated with the Σ-type, the discourse referents introduced in the antecedent in (2) can be successfully passed to the subsequent discourse. An advantage of dependent type theory over previous dynamic theories is that such an externally dynamic character of quantification can be captured without any further stipulation; Σ-types and Π-types, which are natural generalizations of existential and universal quantifiers in predicate logic, are equipped with the mechanism to handle dynamic aspects of discourse interpretations.

The work by Sundholm and Ranta[2] provides a foundation for applying the expressiveness of dependent types to problems in natural language discourse interpretation such as donkey anaphora. However, a problem remains: how can the semantic representation in (2) be systematically obtained from the sentence in (1)? From the viewpoint of standard compositional semantics, the problem can be divided into two tasks. The first is to deterministically map the sentence (1) into an

[2] Sundholm (1986) and Ranta (1994) only consider the so-called strong reading of donkey sentences. There are some later works using dependent types that treat other phenomena discussed in the dynamic semantics literature, in particular, Sundholm (1989) for the proportion problem. See also Tanaka *et al.* (2014) and Tanaka (2014) for discussion of Sundholm's analysis of donkey anaphora and treatment of weak and strong readings within the framework of DTS.

underspecified representation, a semantic representation that contains an underspecified element corresponding to the pronoun in question. The second task is to resolve anaphora. In our example, the underspecified element needs to be resolved to $\pi_1\pi_2 u$. The semantics satisfying the requirement of compositionality must provide an explicit procedure for these two tasks.

Dependent Type Semantics (Bekki 2014) provides such a procedure. To give an explicit compositional mapping from sentences to semantic representations, we adopt Combinatory Categorial Grammar (CCG, Steedman 2000) as a syntactic framework. Note that, as emphasized in Bekki (2014), DTS can be combined with other categorial grammars; see Kubota and Levine (2017) for a concrete proposal that combines DTS with a type-logical grammar. In compositional mapping, an anaphoric expression is mapped on to an underspecified element. The process of resolving underspecification is formulated as the process of *type checking*. Using the machinery of underspecified semantics in DTS, we will give an analysis of presuppositions in Section 2.3.

2.2 *Common nouns: types or predicates?*

There are two possible approaches to representing basic sentences like *A man entered* in dependent type theory. One is the approach proposed in Ranta (1994) and Luo (2012a,b), according to which common nouns like *man* are interpreted as *types* so that the sentence is represented as (3) in our notation.

(3) $\begin{bmatrix} x : \mathbf{man} \\ \mathbf{enter}(x) \end{bmatrix}$

One problem with this approach is that it is not straightforward to analyze *predicational* sentences, i.e., sentences containing *predicate nominals*, such as (4a, b).[3]

(4) a. John is *a man*.
b. Bob considers Mary *a genius*.

One might analyze (4a) as a judgement **john** : **man**. However, a judgement itself can neither be negated nor embedded under a log-

[3] See Mikkelsen (2011) for a useful overview of the syntax and semantics of predicational sentences.

ical operator. Accordingly, it is not clear how to account for the fact that a predicational sentence can be negated, as in (5a), or appear in the antecedent of a conditional, as in (5b).

(5) a. John is not a man.

b. If John is a man, ...

Nor is it clear how to analyze a construction embedding a predicational sentence as in (4b).

One might try to analyze *be*-verbs as the so-called "*is*-of identity" along Russell-Montague lines (Russell 1919; Montague 1973). This enables us to represent (4a) as a *proposition*, as in (6), rather than as a judgement.

(6) $$\begin{bmatrix} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{bmatrix}$$

Then, (5a) and (5b) can be represented as follows:

(7) a. $$\neg\begin{bmatrix} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{bmatrix}$$

b. $$\begin{bmatrix} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{bmatrix} \rightarrow \cdots$$

There are two problems with this approach, however. First, this analysis predicts that the predicate nominal *a man* introduces a discourse referent in terms of Σ-types. Contrary to this prediction, a predicate nominal cannot serve as an antecedent of an anaphoric pronoun such as *he* or *she* (Kuno 1970; Mikkelsen 2005); hence it does not introduce an individual discourse referent.[4]

The second problem is the interpretation of equality. In dependent type theory, equality is relativised to some type A and the formation rule requires the arguments of equality symbols to have type A:

[4] The form of the pronoun anaphoric on a predicate nominal in (i) must be *it*, rather than *him*; the relative pronoun in (ii) must be *which*, not *who* (Kuno 1970; Mikkelsen 2005).

(i) He is a fool, although he doesn't look { it /*him }.

(ii) He is a gentleman, {which / *who} his brother is not.

See Fara (2001) for more discussion of the problems of the Russell-Montague analysis of predicate nominals.

(8) $$\frac{A : \mathbf{type} \quad t : A \quad u : A}{t =_A u : \mathbf{type}} =F$$

Accordingly, the proposition $\mathbf{john} =_{\mathbf{man}} x$ is well-formed only if $\mathbf{john} : \mathbf{man}$ is provable. This is also the case if the proposition is embedded under a logical operator. It thus follows that under the Russell-Montague analysis combined with the equality rule (8), not only the positive sentence (6), but also the negation (7a) and the conditional (7b) presuppose that John is a man. To rescue the common-nouns-as-types view from this problem, one has to provide a more complex analysis of logical operators such as negation and implication.[5] However, the resulting theory would then become more complicated.

As an alternative approach, we interpret a common noun as a predicate. Common nouns in argument position and in predicate position are both analyzed as predicates of type **entity** → **type**.

(9) A man walks. $$\begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{man}(x) \end{bmatrix} \\ \mathbf{walk}(\pi_1 u) \end{bmatrix}$$

(10) John is a man. $\mathbf{man}(\mathbf{john})$

This approach is in line with the traditional analysis of common nouns, so we can integrate standard assumptions in formal semantics into our framework. Moreover, since predicates do not introduce discourse referents, we can explain the impossibility of referential anaphora to predicate nominals.

Retoré (2014) suggests that common nouns can be interpreted both as types and as predicates; for instance, using type **entity**, the common noun *animal* interpreted as a type **animal** could be related to a predicate $\mathbf{animal}^*$ of type **entity** → **type**, via some suitably defined mapping $(\cdot)^*$ from one to another. The question of whether a type system for natural language semantics needs to be enriched with the structures of common nouns would ultimately depend on the treatment of the lexical semantic phenomena it attempts to capture, such as coercion and selectional restriction – phenomena that have been widely discussed in the recent literature on type-theoretical semantics (Asher 2011; Asher and Luo 2012; Bekki and Asher 2013; Retoré

[5] Some discussion of the treatment of negation in the context of dependent type theory can be found in Chatzikyriakidis and Luo (2014).

2014; Kinoshita *et al.* 2016). But investigating this matter further is beyond the scope of the present paper and we leave it to a future study.

2.3 *Analysis of presupposition in DTS*

To handle anaphora and presupposition in a compositional setting, DTS extends dependent type theory with a mechanism of context passing and underspecified terms.

Dependent Type Semantics distinguishes two kinds of propositions: *static* and *dynamic* propositions. Following the Curry-Howard correspondence, we call an object of type **type** (i.e., the type of types) a *static* proposition. A *dynamic* proposition is a function which maps a proof term of the static proposition representing the preceding discourse to a static proposition. The basic idea is that for each (static) proposition P, the information obtained up to that point is passed to P as a proof term. Such a proof term is called a *local context*.

Dependent Type Semantics extends the syntax of dependent type theory with an underspecified term $@_i$, which is used to represent anaphora and presupposition triggers.[6] We show how it can provide a compositional account of anaphora and presupposition. We take the existence presupposition triggered by a definite description as a representative example. Consider the following example.

(11) The book arrived.

The definite description *the book* here triggers the presupposition that there is a book.[7] We analyze the determiner *the* appearing in the subject position as having the CCG category $(S/(S\backslash NP))/N$, and give a semantic representation by using an underspecified term. The lexical

[6] Bekki (2014) provides an overview and comparison of previous approaches to representing underspecification in the context of dependent type theory (Dávila-Pérez 1995; Krahmer and Piwek 1999; Piwek and Krahmer 2000).

[7] Here we take it that the uniqueness presupposition is not part of the conventional meaning of a definite description but can be derived on pragmatic considerations along the lines of Heim (1982). Although it is technically possible to take the uniqueness implication as part of presupposition, the proof-search procedure to find the antecedent of an underspecified term would then become much more complicated.

$N^\bullet = \textbf{entity} \to \delta \to \textbf{type}$	$N^\bullet_{+c} = \textbf{type} \to \delta \to \textbf{type}$
$NP^\bullet = \textbf{entity}$	$NP^\bullet_{+c} = \delta \to \textbf{type}$
$S^\bullet = \delta \to \textbf{type}$	$\overline{S}^\bullet = \delta \to \textbf{type}$
$(C_1/C_2)^\bullet = (C_1\backslash C_2)^\bullet = C_2^\bullet \to C_1^\bullet$	

Figure 2: Mapping syntactic categories to semantic types[9]

entry for *the* can be specified as follows (a mapping $(\cdot)^\bullet$ from syntactic categories to semantic types can be defined as in Figure 2).[8]

(12) the; $(S/(S\backslash NP))/N$; $\lambda n.\lambda v.\lambda c.v\left(\pi_1\left(@_i c :: \begin{bmatrix} x : \textbf{entity} \\ nxc \end{bmatrix}\right)\right)c$

Determiner *the* denotes a function that takes a predicate n denoted by a restrictor and a predicate v denoted by a verb and returns a dynamic proposition, which is in turn a function from a local context c to a (static) proposition. The local context c is passed to the underspecified term $@_i$ as an argument. It is also sent to the predicates n and v as an extra argument, because n and v may contain underspecified terms.

The form $M :: A$ is called *type annotation* and specifies that the term M has type A. When an underspecified term $@_i$ is annotated with a type A, that is, when we have $@_i :: A$, the annotated type A represents the presupposition triggered by this underspecified term. In (12), the underspecified term with a local context, $@_i c$, is annotated with a Σ-type. This means that the underspecified term $@_i$ is a function that takes a local context c as an argument and returns a term having the annotated Σ-type. Given this type annotation, we see that $@_i c$ is a pair of an entity x and a proof term for the proposition that x satisfies

[8] For the purpose of concreteness, we use a type-raised form of semantic representations for determiners. The entry for the determiner *the* in the object position can be given as follows:

the; $((S\backslash NP)\backslash((S\backslash NP)/NP))/N$; $\lambda n.\lambda v.\lambda x.\lambda c.v\left(\pi_1\left(@_i c :: \begin{bmatrix} y : \textbf{entity} \\ nyc \end{bmatrix}\right)\right)xc$

Although there are other possible syntactic analyses of determiners in object position (cf. Bekki 2014), this entry would ensure a concise derivation tree for semantic composition.

[9] Subscripted $+c$ is a syntactic feature for content noun. We represent N_{-c} and NP_{-c} simply as N and NP, respectively. We also abbreviate other syntactic features, which are not relevant to the discussion in this paper.

the predicate n given a local context c. In other words, the annotated type represents the existence presupposition triggered by the definite article. What is applied to the main predicate appearing in v is the first projection of the obtained pair, i.e., a term of type **entity**.

The semantic representation for (11) is derived as follows.[10]

(13)

$$\dfrac{\dfrac{\dfrac{\text{The}}{\begin{array}{c}(S/(S\backslash NP))/N\\ \lambda n.\lambda v.\lambda c.v\left(\pi_1\left(@_1c :: \left[\begin{array}{l}x : \mathbf{entity}\\ nxc\end{array}\right]\right)\right)c\end{array}} \quad \dfrac{\text{book}}{\begin{array}{c}N\\ \lambda x.\lambda c.\mathbf{book}(x)\end{array}}}{\begin{array}{c}S/(S\backslash NP)\\ \lambda v.\lambda c.v\left(\pi_1\left(@_1c :: \left[\begin{array}{l}x : \mathbf{entity}\\ \mathbf{book}(x)\end{array}\right]\right)\right)c\end{array}}> \quad \dfrac{\text{arrived}}{\begin{array}{c}S\backslash NP\\ \lambda x.\lambda c.\mathbf{arrive}(x)\end{array}}}{\begin{array}{c}S\\ \lambda c.\mathbf{arrive}\left(\pi_1\left(@_1c :: \left[\begin{array}{l}x : \mathbf{entity}\\ \mathbf{book}(x)\end{array}\right]\right)\right)\end{array}}>$$

The underspecified term is indexed by a natural number i and each number assigned to an underspecified term is mutually distinct. In the above derivation, an underspecified term introduced by *the* has index 1. Here we assume that the predicate **book** is not context-sensitive so that vacuous abstraction is involved as in $\lambda x \lambda c.\mathbf{book}(x)$; in such a case, the input local context c is simply discarded in the body of the semantic representation. As shown here, the term $@_1c$ in the final representation is annotated with a Σ-type corresponding to the proposition that there is a book. In this way, the annotated type represents the existence presupposition triggered by the definite description *the book*.

[10] In CCG derivation trees, we use two standard combinatory rules: forward (>) and backward (<) function application rules.

$$\frac{X/Y : m \quad Y : n}{X : mn} > \qquad \frac{Y : n \quad X\backslash Y : m}{X : mn} <$$

For instance, the combinatory rule (>) means that an expression having a syntactic category X/Y and a meaning m, combined with an expression having a syntactic category Y and a meaning n, yields an expression having a category X and a meaning mn. Each meaning is represented as a lambda term. See Steedman (2000) for more details.

The resolution of an underspecified term in a semantic representation A amounts to checking that A is well-typed in a given context. More specifically, it is triggered by the following:

(14) $\Gamma, \delta : \textbf{type} \vdash A : \delta \to \textbf{type}$

Here, Γ is a set of assumptions, called a *global* context, which represents the background knowledge; δ is the type of a local context (representing the previous discourse); $A : \delta \to \textbf{type}$ in the consequence shows that A is a dynamic proposition in DTS, that is, a function mapping a given local context of type δ to a static proposition; (14) reflects the requirement that the semantic representation of a (declarative) sentence, i.e., a static proposition, must be of type **type**. This requirement is called the *felicity condition* of a sentence in DTS.

In the case of (11), the resolution process is launched by the following judgement:

(15) $\Gamma, \delta : \textbf{type} \vdash \lambda c.\,\textbf{arrive}\left(\pi_1\left(@_1 c :: \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}\right)\right) : \delta \to \textbf{type}.$

Assuming that $\textbf{arrive} : \textbf{entity} \to \textbf{type}$ is in the global context Γ, the type of $@_1$ is determined by the following derivation.[11]

(16)

$$
\dfrac{
  \dfrac{}{\textbf{arrive} : \textbf{entity} \to \textbf{type}}\,(CON)
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        @_1 : \delta \to \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}
        \qquad
        \overline{c : \delta}^{\,1}
      }{
        @_1 c : \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}
      }\,(\Pi E)
    }{
      \left(@_1 c :: \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}\right) : \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}
    }\,(ann)
  }{
    \pi_1\left(@_1 c :: \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}\right) : \textbf{entity}
  }\,(\Sigma E)
}{
  \dfrac{
    \textbf{arrive}\left(\pi_1\left(@_1 c :: \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}\right)\right) : \textbf{type}
  }{
    \lambda c.\,\textbf{arrive}\left(\pi_1\left(@_1 c :: \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}\right)\right) : \delta \to \textbf{type}
  }\,(\Pi I), 1
}\,(\Pi E)
$$

[11] Here we make use of the following rule, which ensures that one can obtain the annotated term $t :: A$ of type A from any term $t : A$.

$$\dfrac{t : A}{(t :: A) : A}\,(ann)$$

See also the Appendix for other derivation rules.

The open branch of the derivation, repeated in (17), requires that in order for the semantic representation in question to be well-typed, one has to construct a proof term for the proposition that there is a book. (This is due to the @-formation rule. See Definition 12 in the Appendix.)

(17) $@_1 : \delta \to \begin{bmatrix} x : \textbf{entity} \\ \textbf{book}(x) \end{bmatrix}$

In other words, if one assumes that (11) is a felicitous utterance, the proposition that there is a book must be true. This requirement corresponds to the existence presupposition of (11).

At the final stage of presupposition resolution, a proof search is carried out to prove (17) and the underspecified term $@_1$ is replaced by the constructed term. More specifically, the process of anaphora/presupposition resolution is defined as follows (Bekki 2014).

(18) Suppose that $\Gamma \vdash @_i : A$ and $\Gamma \vdash M : A$, where Γ is a global context and A is a type. Then a resolution of $@_i$ by M under the context Γ is an equation $@_i =_A M$.

In the example considered here, if a proof term for the presupposition that there is a book is constructed, it can replace the underspecified term $@_1$. Such a proof construction is possible when, for instance, *the book* appears in contexts as shown in (19a, b).

(19) a. If John ordered a book last week, the book will arrive today.
b. John ordered a book last week and the book arrived today.

In general, if S' entails the presuppositions of S, constructions such as S' and S and If S' then S do not inherit the presuppositions of S. In such a case, it is said that the presupposition is *filtered*.

In DTS, examples such as (19a, b) can be handled in the following way. First, the (somewhat simplified) semantic representation for (19a) is derived as shown in (20). The type checking derivation for the final representation of (20) is shown in (21). This derivation specifies the type of $@_1$. We can see that what is required for the representation to be well-typed is to find a term substituted for $@_1$ in (22).

(20)

$$
\dfrac{
\dfrac{\text{If}}{\begin{array}{c} S/S/S \\ \lambda p.\lambda q.\lambda c.\,(v\!:pc)\rightarrow q(c,v)\end{array}}
\qquad
\dfrac{
\dfrac{\text{John}}{\begin{array}{c} NP \\ \mathbf{john}\end{array}}
\qquad
\dfrac{
\dfrac{\text{ordered}}{\begin{array}{c}(S\backslash NP)/NP \\ \lambda y.\lambda x.\lambda c.\,\mathbf{order}(x,y)\end{array}}
\qquad
\dfrac{
\dfrac{\text{a}}{\begin{array}{c}(S\backslash NP)\backslash((S\backslash NP)/NP)/N \\ \lambda n.\lambda v.\lambda x.\lambda c.\begin{bmatrix} u:\begin{bmatrix} y:\mathbf{entity}\\ nyc\end{bmatrix}\\ v(\pi_1 u)x(c,u)\end{bmatrix}\end{array}}
\qquad
\dfrac{\text{book}}{\begin{array}{c} N \\ \lambda x.\lambda c.\,\mathbf{book}(x)\end{array}}
}{\begin{array}{c}(S\backslash NP)\backslash((S\backslash NP)/NP) \\ \lambda v.\lambda x.\lambda c.\begin{bmatrix} u:\begin{bmatrix} y:\mathbf{entity}\\ \mathbf{book}(y)\end{bmatrix}\\ v(\pi_1 u)x(c,u)\end{bmatrix}\end{array}}>
}{\begin{array}{c} S\backslash NP \\ \lambda x.\lambda c.\begin{bmatrix} u:\begin{bmatrix} y:\mathbf{entity}\\ \mathbf{book}(y)\end{bmatrix}\\ \mathbf{order}(x,\pi_1 u)\end{bmatrix}\end{array}}<
}{\begin{array}{c} S \\ \lambda c.\begin{bmatrix} u:\begin{bmatrix} y:\mathbf{entity}\\ \mathbf{book}(y)\end{bmatrix}\\ \mathbf{order}(\mathbf{john},\pi_1 u)\end{bmatrix}\end{array}}<
}{\begin{array}{c} S/S \\ \lambda q.\lambda c.\left(v\!:\begin{bmatrix} u:\begin{bmatrix} y:\mathbf{entity}\\ \mathbf{book}(y)\end{bmatrix}\\ \mathbf{order}(\mathbf{john},\pi_1 u)\end{bmatrix}\right)\rightarrow q(c,v)\end{array}}>
\qquad
\dfrac{\text{the book will arrive}}{\begin{array}{c} S \\ \mathbf{arrive}\left(\pi_1\left(@_1 c::\begin{bmatrix} x:\mathbf{entity}\\ \mathbf{book}(x)\end{bmatrix}\right)\right)\end{array}}
}{
\lambda c.\left(v\!:\begin{bmatrix} u:\begin{bmatrix} x:\mathbf{entity}\\ \mathbf{book}(x)\end{bmatrix}\\ \mathbf{order}(\mathbf{john},\pi_1 u)\end{bmatrix}\right)\rightarrow \mathbf{arrive}\left(\pi_1\left(@_1(c,v):\begin{bmatrix} x:\mathbf{entity}\\ \mathbf{book}(x)\end{bmatrix}\right)\right)
}>
$$

(21)

$$
\begin{prooftree}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{$\begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} : \mathbf{type}$}
\AxiomC{}
\UnaryInfC{$\begin{array}{c} \mathbf{arrive} \\ : \mathbf{entity} \to \mathbf{type} \end{array}$}
\AxiomC{$@_1 : \begin{bmatrix} \delta \\ \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}$}
\AxiomC{}
\RightLabel{2}
\UnaryInfC{$c : \delta$}
\AxiomC{}
\RightLabel{1}
\UnaryInfC{$t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}$}
\RightLabel{(ΣI)}
\BinaryInfC{$(c, t) : \begin{bmatrix} \delta \\ \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} \end{bmatrix}$}
\RightLabel{(ΠE)}
\BinaryInfC{$@_1(c, t) : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}$}
\RightLabel{(ann)}
\UnaryInfC{$\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right) : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}$}
\RightLabel{(ΣE)}
\UnaryInfC{$\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right) : \mathbf{entity}$}
\RightLabel{(ΠE)}
\BinaryInfC{$\mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \mathbf{type}$}
\RightLabel{$(\Pi F), 1$}
\BinaryInfC{$\left(t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}\right) \to \mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \mathbf{type}$}
\RightLabel{$(\Pi I), 2$}
\UnaryInfC{$\lambda c. \left(t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}\right) \to \mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \delta \to \mathbf{type}$}
\end{prooftree}
$$

(22) $$@_1 : \left[\begin{array}{l} \delta \\ \left[\begin{array}{l} u : \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right] \\ \textbf{order}(\textbf{john}, \pi_1 u) \end{array}\right] \end{array}\right] \to \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]$$

In this case, without using the information provided in the previous discourse in δ, a proof of the proposition that there is a book can be obtained from the antecedent of the conditional; one can find a term that can replace $@_1$, namely, $\lambda c.\pi_1\pi_2 c$.[12] By replacing $@_1$ with the constructed term $\lambda c.\pi_1\pi_2 c$ in the representation given in (20), we can eventually obtain the following semantic representation for (19a), which captures the intended reading.

(23) $$\left(t : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right] \\ \textbf{order}(\textbf{john}, \pi_1 u) \end{array}\right] \right) \to \textbf{arrive}(\pi_1\pi_1 t)$$

Another well-known characteristic property of a presupposition is that it *projects* out of embedded contexts such as negation and the antecedent of a conditional. Thus, not only the positive sentence (11) but also the negated sentence (24a) and the antecedent of a conditional (24b) imply that there is a book.

(24) a. The book didn't arrive. NEGATION

b. If the book arrives, Susan will be happy. CONDITIONAL

In DTS, (24a, b) can be given the following semantic representations.

(25) a. $$\lambda c.\neg\textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)$$

b. $$\lambda c.\textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right) \to \textbf{happy}(\textbf{susan})$$

It can be shown that for the semantic representations (25a) and (25b) to be well-typed, it is required to find a proof term for the proposition that there is a book. Thus, in order to prove that (25b) has type $\delta \to \textbf{type}$, one has to prove that the antecedent is of type **type** under the given local context c. Since the antecedent in (25b) corresponds to the proposition that the book arrived, this yields the derivation that

[12] When such a filtration does not occur, the entire sentence can have a presupposition that is resolved by the information in δ (i.e., the information in the previous discourse) or by the information in the global context (background knowledge).

contains the type checking process in (16) as a sub-derivation. Accordingly, it is correctly predicted that (24b) has the same existence presupposition as the simple sentence in (11). Note that, in dependent type theory, the negation $\neg A$ is defined using the implication $A \to \bot$, which in turn is a degenerate form of a Π-type. Thus, the same explanation applies to the case of negation in (24a) as well. In this way, we can explain basic projection patterns of presuppositions within the framework of DTS.[13]

Before moving on to the case of factive presuppositions, let us mention one feature of underspecified terms in DTS; that is, an underspecified term can occur inside a type annotation of another underspecified term. This feature enables us to handle *nested* presuppositions. As a typical example, consider a definite noun phrase such as *the book that he wrote*. Omitting the details of compositional derivation, we can assign the following semantic representation to this complex NP.

(26) $$\lambda v.\lambda c.v\left(\pi_1\left(@_1 c :: \begin{bmatrix} x : \mathbf{entity} \\ \begin{bmatrix} \mathbf{book}(x) \\ \mathbf{write}(@_2 c :: \mathbf{entity}, x) \end{bmatrix} \end{bmatrix}\right)\right)c$$

Here, the underspecified term $@_2$, introduced by the pronoun *he*, occurs inside the type annotation of the underspecified term $@_1$ introduced by *the*.[14] In this case, one can resolve the most embedded underspecified term $@_2$ first and then resolve the outer underspecified term $@_2$ subsequently. A more detailed discussion of nested presupposition is given in Bekki and Mineshima (2017).

3 ANALYZING FACTIVITY IN DTS

In this section, we provide an analysis of factive predicates in DTS. We take the verb *know* as a representative of a factive predicate and provide its semantic representation. We start by summarizing some semantic properties of *know*, in comparison with the non-factive verb *believe*.

[13] Bekki and Satoh (2015) provide a definition for decidable fragment of dependent type theory with an underspecified term and formulate its type-checking algorithm. They also provide an implementation of the algorithm.

[14] It is also possible to add gender information associated with personal pronouns as a presupposition. See Bekki and Mineshima (2017).

3.1 *Inferences with factive and non-factive verbs*

The factive verb *know* and the non-factive verb *believe* show different inference patterns with respect to the form of complements they take. In what follows, we will focus on two types of complements, declarative complements and NP-complements.

Consider the examples in (27), where the verbs *know* and *believe* take a declarative complement.

(27) a. John knows that Mary is successful.

b. John believes that Mary is successful.

(28) Mary is successful.

(27a) implies (28), while (27b) does not. In the context of epistemic logic (Hintikka 1962; Meyer and van der Hoek 2004), the inference from (27a) to (28) has usually been treated as an instance of entailment (hereafter, we use the notion "entailment" in the sense of logical entailment). In the linguistics literature, by contrast, it has been widely agreed that the inference from (27a) to (28) is not an entailment but a presupposition (Kiparsky and Kiparsky 1970; Beaver 2001), as witness examples in (29) and (30).

(29) a. John does not know that Mary is successful. NEGATION

b. If John knows that Mary is successful, ... CONDITIONAL

(30) a. If Mary is successful, John knows that she is.

b. Mary is successful, and John knows that she is.

The examples in (29a, b) show that the proposition in (28) projects out of the embedded contexts; the examples in (30a, b) shows the filtering of presupposition. Because the antecedent of (30a) or the first conjunct of (30b) entails (28), the sentences do not inherit the presupposition of *know*, in a similar way to (19a, b).

Another interesting difference between *know* and *believe* is shown in (31), where they take an NP-complement of the form *the N that P*.

(31) a. John believes the rumor that Mary came.
$\Rightarrow$ John believes that Mary came.

b. John knows the rumor that Mary came.
$\not\Rightarrow$ John knows that Mary came.

Figure 3: Entailments (⇒) and presuppositions (▷) associated with factive and non-factive verbs (N refers to a non-veridical content noun)

K1	x knows that P	▷	P
K2	x knows the N that P	$\not\Rightarrow$	x knows that P
K3	x knows the N that P	▷	There is a N that P
B1	x believes that P	$\not\Rightarrow$	P
B2	x believes the N that P	⇒	x believes that P
B3	x believes the N that P	▷	There is a N that P

The non-factive verb *believe* licenses the inference from x *Vs the* N *that* P to x *Vs that* P, where N is a (non-veridical) content noun, such as *rumor*, *story*, and *hypothesis*, that takes a propositional complement; by contrast, the factive verb *know* does not license this pattern of inference (Vendler 1972; Ginzburg 1995a,b; Uegaki 2016).

Figure 3 shows a summary of the inference patterns for *know* and *believe* that we are concerned with in this paper. A remark is in order regarding the non-entailment in K2. There is a class of content nouns that does not follow the pattern in K2. A typical example is the content noun *fact*; "x knows the fact that P" entails "x knows that P", and vice versa. We call this class of nouns *veridical* content nouns and distinguish them from *non-veridical* content nouns such as *rumor* and *story*. The inference pattern in K2 only applies to non-veridical content nouns. We discuss the case of veridical content nouns at the end of Section 3.3.

To predict these inference patterns in a compositional setting, one needs to provide an adequate account of the lexical semantic difference between factive and non-factive verbs. One possible approach is to consider the two types of verbs select for different semantic objects. More specifically, it has been proposed by a number of authors that the non-factive verb *believe* selects for a *proposition*, whereas the factive verb *know* selects for a *fact* (Vendler 1972; Parsons 1993; Ginzburg 1995a,b; King 2002). In the next section, we will explore such a semantic analysis of factive and non-factive verbs within the framework of DTS.

3.2 *Declarative complements*

We treat factive and non-factive verbs as predicates having different semantic types. We analyze the non-factive verb *believe* as taking two arguments, a term of type **entity** and a proposition. In our notation, the predicate **believe** has the following type:[15]

(32) $\mathbf{believe} : \mathbf{entity} \to \mathbf{type} \to \mathbf{type}$

By contrast, we analyze the factive verb *know* as taking three arguments: (i) an entity representing the agent, (ii) a proposition that serves as the content of knowledge, and (iii) a proof term of that proposition. The predicate **know** has the following semantic type:

(33) $\mathbf{know} : \mathbf{entity} \to (\mathsf{P} : \mathbf{type}) \to \mathsf{P} \to \mathbf{type}.$

As mentioned in Section 2.1, the existence of a proof term a of type P corresponds to the truth of proposition P. One may read $\mathbf{know}(x)(\mathsf{P})(a)$ as *the agent x obtains evidence a of the proposition* P.

The standard analysis of *know* in formal semantics follows Hintikka's (1969) possible world semantics, which fails to capture the notion of evidence or justification that has been traditionally associated with the concept of knowledge. An advantage of dependent type theory is that it is equipped with proofs as first-class objects and thus enables us to analyze the factive verb *know* as a predicate over a proof (evidence) of a proposition. Our analysis is also compatible with Vendler's view that *know* and *believe* select for different semantic objects. Note that, in our approach, the notion of facts is not taken as primitive but analyzed in terms of the notion of evidence of a proposition.

The idea that a proof term of a proposition serves as an antecedent of anaphor can be traced back to Ranta (1994), where under the assumption that proofs are identified with *events* it is claimed that aspectual verbs like *stop* presuppose the existence of a proof. Also, Krahmer and Piwek (1999) briefly mentioned that the presuppositions triggered by noun phrases like *the fact that P* can be treated in a similar way (see also Section 3.3 for some discussion). Our claim is that the idea that proof terms act as antecedents of anaphora can be applied to the presuppositions of factive verbs in general.

[15] We leave open the possibility of decomposing the semantic representation of belief sentences in terms of possible worlds. See Tanaka *et al.* (2015) for discussion in the context of DTS. The problem of opacity and hyperintensionality (Fox and Lappin 2005) is beyond the scope of the present paper.

To account for the presuppositional inferences summarized in Section 3.1, we use the following lexical entry for *know*.[16]

(34) know; $(S\backslash NP)/\overline{S}$; $\lambda p.\lambda x.\lambda c.\mathbf{know}(x)(pc)(@_i c)$

Here the argument p is a dynamic proposition expressed by the declarative complement of *know*. The underspecified term $@_i$ takes a local context c as an argument and requires one to construct a proof term of type pc, i.e., to find *evidence* of the (static) proposition pc being true. If such a proof term is constructed, it fills the third argument position of the predicate **know**. In sum, the sentence x *knows that* P presupposes that there is a proof (evidence) of P and asserts that the agent x obtains it, i.e., x has a proof (evidence) of the proposition P.

Let us illustrate with (27a) how to give a compositional analysis of a construction containing *know*. The semantic representation for (27a) is given by the following CCG derivation tree.

(35)

$$\dfrac{\dfrac{\text{John}}{\begin{array}{c}NP\\ \mathbf{john}\end{array}} \quad \dfrac{\dfrac{\text{knows}}{\begin{array}{c}(S\backslash NP)/\overline{S}\\ \lambda p.\lambda x.\lambda c.\mathbf{know}(x)(pc)(@_1 c)\end{array}} \quad \dfrac{\dfrac{\text{that}}{\begin{array}{c}\overline{S}/S\\ \lambda \mathrm{P}.\mathrm{P}\end{array}} \quad \dfrac{\text{Mary is successful}}{\begin{array}{c}S\\ \lambda c.\mathbf{successful}(\mathbf{mary})\end{array}}}{\begin{array}{c}\overline{S}\\ \lambda c.\mathbf{successful}(\mathbf{mary})\end{array}}>}{\begin{array}{c}S\backslash NP\\ \lambda x.\lambda c.\mathbf{know}(x)(\mathbf{successful}(\mathbf{mary}))(@_1 c)\end{array}}>}{\begin{array}{c}S\\ \lambda c.\mathbf{know}(\mathbf{john})(\mathbf{successful}(\mathbf{mary}))(@_1 c)\end{array}}<$$

Then, the derivation (36) checks whether the semantic representation is well-typed.The open branch ending up with $\delta \to \mathbf{successful}(\mathbf{mary})$ shows the presupposition of this representation, which is the factive presupposition of (27a). In this way, we can correctly predict that the presuppositional inference from (27a) to (28) holds. The inference mechanism we described in Section 2.3 for the existence presupposition of definite descriptions can be extended for the case of *know*. In particular, it is easy to see that the projection inference in (29) and the filtering inference in (30) can be accounted for in the same way as those in (24) and (19), respectively.

[16] In (34), the underspecified term $@_i c$ is not annotated with its type pc, since it is inferable from the type of the predicate **know**.

(36)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\begin{array}{c}\textbf{know}\\ :\textbf{entity}\to(\mathsf{P}:\textbf{type})\\ \to\mathsf{P}\to\textbf{type}\end{array}}\,(CON)\qquad\dfrac{}{\textbf{john}:\textbf{entity}}\,(CON)}{\begin{array}{c}\textbf{know}(\textbf{john})\\ :(\mathsf{P}:\textbf{type})\to\mathsf{P}\to\textbf{type}\end{array}}\,(\Pi E)\qquad\dfrac{\dfrac{}{\begin{array}{c}\textbf{successful}\\ :\textbf{entity}\to\textbf{type}\end{array}}\,(CON)\qquad\dfrac{}{\textbf{mary}:\textbf{entity}}\,(CON)}{\textbf{successful}(\textbf{mary}):\textbf{type}}\,(\Pi E)}{\begin{array}{c}\textbf{know}(\textbf{john})(\textbf{successful}(\textbf{mary}))\\ :\textbf{successful}(\textbf{mary})\to\textbf{type}\end{array}}\,(\Pi E)\qquad\dfrac{\begin{array}{c}@_1\\ :\delta\to\textbf{successful}(\textbf{mary})\end{array}\qquad\dfrac{}{c:\delta}\,1}{\begin{array}{c}@_1c\\ :\textbf{successful}(\textbf{mary})\end{array}}\,(\Pi E)}{\dfrac{\textbf{know}(\textbf{john})(\textbf{successful}(\textbf{mary}))(@_1c):\textbf{type}}{\lambda c.\,\textbf{know}(\textbf{john})(\textbf{successful}(\textbf{mary}))(@_1c):\delta\to\textbf{type}}\,(\Pi I),1}\,(\Pi E)$$

It is known that a sentence such as (37) poses the so-called *binding problem* (Karttunen 1971; Karttunen and Peters 1979; Cooper 1983).

(37) A student regrets that she talked.

Here, the existential quantification introduced by *a student* binds the pronoun *she* that appears in the presupposed content. This is an instance of the nested presuppositions that we mentioned in Section 2.3. In DTS, (37) can be given the semantic representation as shown in (38) (where the semantic representation of *regrets* is analogous to that of *knows* above). In this resulting representation, the Σ-type corresponding to the subject *a student* binds the variable u in the second argument of **regret**, which correponds to the type of $@_2(c, u)$ introduced by the factive presupposition of *regret*. With the help of the type checking procedure, this enables us to capture the dependency between assertion and presupposition.[17]

3.3 *NP-complements*

The analysis presented so far can be extended to the analysis of NP-complements. Let us first take the case of *believe*. Consider the example in (31a). The semantic representation of the definite NP *the rumor that Mary came* appearing in the object position can be derived as in (39). Since *rumor* is a content noun, we treat it as having the syntactic category N_{+c} with the syntactic feature $+c$. Correspondingly, the predicate **rumor** is analyzed as a predicate over propositions; its type is **type** $\rightarrow$ **type**. The semantic representation of definite article *the* is given in the same way as the one in Section 2.3 except that it combines with a predicate over propositions (i.e., objects of type **type**), rather than with a predicate over entities.[18]

The semantic representation for the premise sentence in (31a) can be derived as shown in (40). The resulting representation presupposes that there is a rumor whose content is identified with **come**(**mary**). This is the existence presupposition triggered by the NP-complement

[17] See Bekki and Mineshima (2017) for more details.

[18] Strictly speaking, to combine Σ-types with predicates over propositions requires the notion of type hierarchy (Martin-Löf 1984). For ease of exposition, we refer to the base type ($\mathbf{type}_0$) simply as **type**.

(38)

$$
\dfrac{
\dfrac{
\dfrac{\text{A}}{\begin{array}{c} S/(S\backslash NP)/N \\ \lambda n.\lambda v.\lambda c. \\ \left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ nxc \end{array}\right] \\ v(\pi_1 u)(c,u) \end{array}\right] \end{array}}
\quad
\dfrac{\text{student}}{\begin{array}{c} N \\ \lambda x.\lambda c.\,\mathbf{student}(x) \end{array}}
}{\begin{array}{c} S/(S\backslash NP) \\ \lambda v.\lambda c.\left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{student}(x) \end{array}\right] \\ v(\pi_1 u)(c,u) \end{array}\right] \end{array}}>
\quad
\dfrac{
\dfrac{\text{regrets}}{\begin{array}{c} (S\backslash NP)/\overline{S} \\ \lambda p.\lambda x.\lambda c. \\ \mathbf{regret}(x)(pc)(@_2 c) \end{array}}
\quad
\dfrac{
\dfrac{\text{that}}{\begin{array}{c} \overline{S}/S \\ \lambda p.\,p \end{array}}
\quad
\dfrac{\text{she talked}}{\begin{array}{c} S \\ \lambda c.\,\mathbf{talk}(@_1 c :: \mathbf{entity}) \end{array}}
}{\begin{array}{c} \overline{S} \\ \lambda c.\,\mathbf{talk}(@_1 c :: \mathbf{entity}) \end{array}}>
}{\begin{array}{c} S\backslash NP \\ \lambda x.\lambda c.\,\mathbf{regret}(x)(\mathbf{talk}(@_1 c :: \mathbf{entity}))(@_2 c) \end{array}}>
}{\begin{array}{c} S \\ \lambda c.\left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{student}(x) \end{array}\right] \\ \mathbf{regret}(\pi_1 u)(\mathbf{talk}(@_1(c,u) :: \mathbf{entity}))(@_2(c,u)) \end{array}\right] \end{array}}>
$$

(39)

$$\dfrac{\dfrac{\text{the}}{\begin{array}{c}((S\backslash NP)\backslash((S\backslash NP)/NP_{+c}))/N_{+c}\\ :\lambda n.\lambda v.\lambda x.\lambda c.\\ v\left(\lambda c'.\pi_1\left(@_1 c :: \begin{bmatrix}\mathsf{P} : \mathbf{type}\\ n\mathsf{P}c\end{bmatrix}\right)\right)xc\end{array}} \qquad \dfrac{\dfrac{\text{rumor}}{\begin{array}{c}N_{+c}\\ \lambda\mathsf{P}.\lambda c.\,\mathbf{rumor}(\mathsf{P})\end{array}} \qquad \dfrac{\dfrac{\text{that}}{\begin{array}{c}(N_{+c}\backslash N_{+c})/S\\ :\lambda p.\lambda n.\lambda\mathsf{P}.\lambda c.\\ \begin{bmatrix}\mathsf{P} =_{\mathbf{type}} pc\\ n\mathsf{P}c\end{bmatrix}\end{array}} \quad \dfrac{\text{Mary came}}{\begin{array}{c}S\\ \lambda c.\mathbf{come}(\mathbf{mary})\end{array}}}{\begin{array}{c}N_{+c}\backslash N_{+c}\\ \lambda n.\lambda\mathsf{P}.\lambda c.\begin{bmatrix}\mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary})\\ n\mathsf{P}c\end{bmatrix}\end{array}}>}{\begin{array}{c}N_{+c}\\ \lambda\mathsf{P}.\lambda c.\begin{bmatrix}\mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary})\\ \mathbf{rumor}(\mathsf{P})\end{bmatrix}\end{array}}<}{\begin{array}{c}(S\backslash NP)\backslash((S\backslash NP)/NP_{+c})\\ \lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\pi_1\left(@_1 c :: \begin{bmatrix}\mathsf{P} : \mathbf{type}\\ \begin{bmatrix}\mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary})\\ \mathbf{rumor}(\mathsf{P})\end{bmatrix}\end{bmatrix}\right)\right)xc\end{array}}>$$

(40)

$$\dfrac{\dfrac{\text{John}}{\begin{array}{c} NP \\ \mathbf{john} \end{array}} \qquad \dfrac{\dfrac{\text{believes}}{\begin{array}{c} (S\backslash NP)/NP_{+c} \\ \lambda p.\lambda x.\lambda c.\,\mathbf{believe}(x)(pc) \end{array}} \quad \dfrac{\text{the rumor that Mary came}}{\begin{array}{c} (S\backslash NP)\backslash((S\backslash NP)/NP_{+c}) \\ \lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\,\pi_1\left(@_1 c :: \left[\begin{array}{l} \mathsf{P} : \mathbf{type} \\ \left[\begin{array}{l} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{array}\right] \end{array}\right]\right)\right) xc \end{array}}}{\begin{array}{c} S\backslash NP \\ \lambda x.\lambda c.\,\mathbf{believe}(x)\left(\pi_1\left(@_i c :: \left[\begin{array}{l} \mathsf{P} : \mathbf{type} \\ \left[\begin{array}{l} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{array}\right] \end{array}\right]\right)\right) \end{array}}<}{\begin{array}{c} S \\ \lambda c.\,\mathbf{believe}(\mathbf{john})\left(\pi_1\left(@_i c :: \left[\begin{array}{l} \mathsf{P} : \mathbf{type} \\ \left[\begin{array}{l} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{array}\right] \end{array}\right]\right)\right) \end{array}}<$$

(cf. B3 in Figure 3). When this presupposition is satisfied, the resulting semantic representation can be reduced to **believe**(**john**, P), where we have P : **type**, P $=_{\mathbf{type}}$ **come**(**mary**), and **rumor**(P). Thus, we can derive the representation **believe**(**john**, **come**(**mary**)), which is the representation for the conclusion in (31a). Hence, we can correctly derive the entailment pattern B2 in Figure 3.

It should be noted that, in the case of (31a), the predicate **rumor** does not contribute to the content of belief. By contrast, (31b) shows that, in the case of the factive verb *know* taking an NP-complement, the predicate **rumor** is part of the content of knowledge ascribed to the agent. The premise sentence in (31b) can be paraphrased as *John knows that there is a rumor that Mary came*. To handle (31b), then, we use the following lexical entries for the non-presuppositional use of *the*, which we refer to by the_{pred}.

(41) the_{pred} (subject position);

$$(S/(S\backslash NP_{+c}))/N_{+c}\,;\ \lambda n.\lambda v.\lambda c.\,v\left(\lambda c'.\begin{bmatrix}\mathsf{P}:\mathbf{type}\\ n\mathsf{P}c\end{bmatrix}\right)c$$

(42) the_{pred} (object position);

$$((S\backslash NP)\backslash((S\backslash NP)/NP_{+c}))/N_{+c}\,;\ \lambda n.\lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\begin{bmatrix}\mathsf{P}:\mathbf{type}\\ n\mathsf{P}c\end{bmatrix}\right)xc$$

In contrast to the entry given in (12), the_{pred} does not have existence presupposition and passes the whole existential proposition (Σ-type) to the main predicate.[19]

We take it that the existence presupposition associated with the premise sentence in (31b) comes from the factive verb *know*. Using the entry in (42), the semantic representation for the premise sentence in (31b) can be derived as in (43). The semantic representation derived in (43) presupposes that there is a rumor that Mary came and asserts that John has evidence for it. This is clearly distinguished from the reading

[19] Such a non-presuppositional use of definite description is also needed to handle examples such as *The king of France does not exist*, where the use of *the* does not presuppose the existence of the king of France.

(43)

$$\dfrac{\dfrac{\text{John}}{\begin{array}{c} NP \\ \mathbf{john} \end{array}} \qquad \dfrac{\dfrac{\text{knows}}{\begin{array}{c} (S\backslash NP)/NP \\ \lambda p.\lambda x.\lambda c.\mathbf{know}(x)(pc)(@_1 c) \end{array}} \qquad \dfrac{\text{the rumor that Mary came}}{\begin{array}{c} (S\backslash NP)\backslash((S\backslash NP)/NP) \\ \lambda v.\lambda x.\lambda c.v\left(\lambda c'.\begin{bmatrix} \mathsf{P}:\mathbf{type} \\ \begin{bmatrix} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{bmatrix} \end{bmatrix}\right)xc \end{array}}}{\begin{array}{c} S\backslash NP \\ \lambda x.\lambda c.\mathbf{know}(x)\left(\begin{bmatrix} \mathsf{P}:\mathbf{type} \\ \begin{bmatrix} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{bmatrix} \end{bmatrix}\right)(@_1 c) \end{array}}<}{\begin{array}{c} S \\ \lambda c.\mathbf{know}(\mathbf{john})\left(\begin{bmatrix} \mathsf{P}:\mathbf{type} \\ \begin{bmatrix} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{rumor}(\mathsf{P}) \end{bmatrix} \end{bmatrix}\right)(@_1 c) \end{array}}<$$

Figure 4:
Inferences associated with veridical content nouns

K4	x knows the fact that P	$\Longleftrightarrow$	x knows that P
K5	x knows the fact that P	$\triangleright$	There is a fact that P
K6	x knows the fact that P	$\triangleright$	P

that John has evidence that Mary came, hence, we can account for the non-entailment in (31b), schematically given as K2 in Figure 3.[20]

As noted in Section 3.1, veridical content nouns such as *fact* and *truth* show a different entailment pattern from non-veridical content nouns such as *rumor* and *story*. The relevant inference patterns are summarized in Figure 4. The present analysis can naturally handle these inference patterns as well. Consider (44):

(44) John knows the fact that Mary came.

In the same way as the derivation in (43), we can obtain the semantic representation for (44):

(45) $$\lambda c.\,\mathbf{know}(\mathbf{john})\left(\left[\begin{array}{l}\mathsf{P}:\mathbf{type}\\ \left[\begin{array}{l}\mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary})\\ \mathbf{fact}(\mathsf{P})\end{array}\right]\end{array}\right]\right)(@_1 c)$$

The underspecified term $@_1$ in (45) triggers the presupposition that there is the fact that Mary came, which accounts for K5 in Figure 4. To account for the other inference patterns, we may posit two axioms. The first axiom is the one concerning the lexical meaning of the veridical content noun *fact*:

(46) **Axiom 1** : $(\mathsf{P}\!:\mathbf{type}) \to (\mathbf{fact}(\mathsf{P}) \leftrightarrow \mathsf{P})$

Using this axiom, one can construct a proof term of **come**(**mary**) from the presupposition in (45), hence K6 in Figure 4 follows.

The second axiom we need is about the closure property of *know*:

(47) **Axiom 2** : $(x\!:\mathbf{entity}) \to (\mathsf{P}\!:\mathbf{type}) \to (\mathsf{Q}\!:\mathbf{type}) \to (a\!:\mathsf{P}) \to$
$\mathbf{know}(x)(\mathsf{P})(a) \to (f\!:\mathsf{P} \to \mathsf{Q}) \to \mathbf{know}(x)(\mathsf{Q})(f\,a)$

The sentence *John knows that Mary came* can be compositionally assigned the semantic representation in (48), in the same way as the derivation shown in (35).

[20] As pointed out by an anonymous reviewer, the current analysis allows the combination of the verb *know* and the presuppositional *the*. This yields an unintended reading for (31b) where it is presupposed that Mary came. Though this undesirable reading could be blocked by involving a complicated syntactic analysis, we consider details of such an analysis as beyond the scope of this paper.

(48) $\lambda c.\mathbf{know}(\mathbf{john})(\mathbf{come}(\mathbf{mary}))(@_2 c)$

It is easy to derive (48) from (45) given an initial context c. First, assume that the presupposition in (45) is satisfied, that is, there is a proof term substituted for $@_1$ in (45). Using Axiom 1, we can construct a proof term substituted for $@_2$ in (48), that is, a proof term of **come**(**mary**), as well as a proof term for the proposition in (49).

(49) $$\begin{bmatrix} \mathsf{P} : \mathbf{type} \\ \begin{bmatrix} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{fact}(\mathsf{P}) \end{bmatrix} \end{bmatrix} \to \mathbf{come}(\mathbf{mary})$$

Hence, applying Axiom 2, we can obtain a proof term of (48). The other direction, i.e., the inference from (48) to (45), is derived in the same manner. Thus we can account for the pattern in K4.

Note that the present analysis of the inference pattern in K6 is different from what is suggested by Krahmer and Piwek (1999). These authors briefly discuss the presupposition triggered by the factive construction *(be) annoyed by the fact that* P. They treat this construction as one complex predicate, assuming that its presupposition is P. In our approach, *know the fact that* P is analyzed not as directly presupposing that P, but as presupposing the existence of the fact whose content is P. Under the present analysis, the inference in K6 is explained in terms of the lexical knowledge concerning the content noun *fact*.

4 CONCLUSION

This paper has attempted to provide an analysis of presuppositions and factivity within the framework of DTS. Under our analysis, factive and non-factive verbs are assigned different semantic types: while the non-factive predicate *believe* selects for a proposition as an object argument, the factive predicate *know* takes a proof-object as an extra argument. Using the machinery of underspecified semantics in DTS, we have illustrated how to account for a variety of inferences concerning factive and non-factive verbs.

Several open issues remain, most notably that of the interpretation of interrogative complements. It is acknowledged in the literature that the factive verb *know* takes interrogative complements, whereas the non-factive verb *believe* does not (Ginzburg 1995a,b; Egré 2008):

(50) a. John {knows, *believes} whether Ann or Bob came.
b. John {knows, *believes} who came.

Providing a detailed analysis of interrogative complements within our proof-theoretic framework is left for another occasion.

ACKNOWLEDGMENT

This paper is a revised and expanded version of a paper presented at the TYpe Theory and LExical Semantics (TYTLES) workshop during the 27th European Summer School in Logic, Language and Information (ESSLLI 2015). We are grateful to Robin Cooper, Christian Retoré, and the audience of TYTLES workshop for helpful discussions. We would also like to thank the three anonymous reviewers of this paper for their valuable comments and suggestions. This work was supported by JST CREST Grant Number JPMJCR1301, Japan. The first author acknowledges the financial support of the JSPS Grant-in-Aid for JSPS Fellows Grant Number 15J11772.

APPENDIX

Definition 1 (Alphabet for $\lambda_{\Pi\Sigma}$) *An alphabet for $\lambda_{\Pi\Sigma}$ is a $\langle Var, Con \rangle$, where Var is a set of variables, and Con is a set of constants. Dependent type semantics employs an alphabet as follows:*

$$\begin{array}{lcl} Var & \overset{def}{\equiv} & \{x, y, z, u, v, ...\} \\ Con & \overset{def}{\equiv} & \{\textbf{entity}, \textbf{book}, \textbf{arrive}, ...\} \end{array}$$

Definition 2 (Preterms) *The collection of preterms is recursively defined as follows (where $x \in Var$ and $c \in Con$, $j = 1, 2$, $i = 0, 1, 2, ...$).*

$$\begin{array}{l} \Lambda := x \mid c \mid @_i \mid \textbf{type}_i \mid \Lambda :: \Lambda \mid (x{:}\Lambda) \to \Lambda \mid \lambda x.\Lambda \mid \Lambda\Lambda \\ \quad \mid \begin{bmatrix} x : \Lambda \\ \Lambda \end{bmatrix} \mid (\Lambda, \Lambda) \mid \pi_j(\Lambda) \mid \Lambda =_\Lambda \Lambda \mid \mathsf{refl}_\Lambda(\Lambda) \mid \mathsf{idpeel}(\Lambda, \Lambda) \\ \quad \mid \bot \mid \top \mid \mathsf{case}_\Lambda(\Lambda_1, ..., \Lambda_n) \end{array}$$

Definition 3 (Signature) *A signature σ is defined recursively as follows.*

$$\sigma := \; () \mid \sigma, c : A$$

where $()$ is an empty signature, $c \in Con$, $A \in \Lambda$ s.t. $\vdash_\sigma A : \textbf{type}_i$ for some $i \in \mathbb{N}$.

Definition 4 (Context) *A context is defined recursively as follows.*

$$\Gamma := \ () \mid \Gamma, x : A$$

where $()$ *is an empty context,* $x \in Var$*,* $A \in \Lambda$ *s.t.* $\Gamma \vdash_\sigma A : \mathbf{type}_i$ *for some* $i \in \mathbb{N}$*.*

Definition 5 (Constant symbol rule) *For any* $(c : A) \in \sigma$*,*

$$\dfrac{}{c : A}\ (CON)$$

Definition 6 (Type rules) *For any* $i \in \mathbb{N}$*,*

$$\dfrac{A : \mathbf{type}_i}{A : \mathbf{type}_{i+1}}\ (\mathbf{type}I) \qquad \dfrac{}{\mathbf{type}_i : \mathbf{type}_{i+1}}\ (\mathbf{type}F)$$

Definition 7 (Π-type) *For any* $i, j \in \mathbb{N}$*,*

$$\dfrac{A : \mathbf{type}_i \quad \begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ B : \mathbf{type}_i\end{array}}{(x{:}A) \to B : \mathbf{type}_i}\ (\Pi F), j \qquad \dfrac{A : \mathbf{type}_i \quad \begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ M : B\end{array}}{\lambda x.M : (x{:}A) \to B}\ (\Pi I), j$$

$$\dfrac{M : (x{:}A) \to B \quad N : A}{MN : B[M/x]}\ (\Pi E)$$

Definition 8 (Σ-type) *For any* $i, j \in \mathbb{N}$*,*

$$\dfrac{A : \mathbf{type}_i \quad \begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ B : \mathbf{type}_i\end{array}}{\begin{bmatrix} x : A \\ B \end{bmatrix} : \mathbf{type}_i}\ (\Sigma F), j \qquad \dfrac{M : A \quad N : B[M/x]}{(M, N) : \begin{bmatrix} x : A \\ B \end{bmatrix}}\ (\Sigma I)$$

$$\dfrac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_1(M) : A}\ (\Sigma E) \qquad \dfrac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_2(M) : B[\pi_1(M)/x]}\ (\Sigma E)$$

Definition 9 (Bottom type) *For any* $i \in \mathbb{N}$*,*

$$\dfrac{}{\bot : \mathbf{type}_0}\ (\bot F) \qquad \dfrac{M : \bot \quad C : \bot \to \mathbf{type}_i}{\mathsf{case}_M() : C(M)}\ (\bot E)$$

Definition 10 (Top type) *For any $i \in \mathbb{N}$,*

$$\frac{}{\top : \mathbf{type}_0}\ (\top F) \qquad \frac{}{() : \top}\ (\top I)$$

$$\frac{M : \top \quad C : \top \to \mathbf{type}_i \quad N : C()}{\mathsf{case}_M(N) : C(M)}\ (\top E)$$

Definition 11 (Id-type) *For any $i \in \mathbb{N}$,*

$$\frac{A : \mathbf{type}_i \quad M : A \quad N : A}{M =_A N : \mathbf{type}_i}\ (\mathsf{Id}F) \qquad \frac{A : \mathbf{type} \quad M : A}{\mathsf{refl}_A(M) : M =_A M}\ (\mathsf{Id}I)$$

$$\frac{E : M_1 =_A M_2 \quad C : (x{:}A) \to (y{:}A) \to (x =_A y \to \mathbf{type}_i) \quad N : (x{:}A) \to Cxx(\mathsf{refl}_A(x))}{\mathsf{idpeel}(e, N) : CM_1M_2E}\ (\mathsf{Id}E)$$

Definition 12 (@-formation rule) *For any $i, j \in \mathbb{N}$,*

$$\frac{A : \mathbf{type}_i \quad A\ true}{@_j : A}\ (@F)$$

Definition 13 (Type annotation rule)

$$\frac{t : A}{(t :: A) : A}\ (ann)$$

REFERENCES

Nicholas ASHER (2011), *Lexical Meaning in Context: A Web of Words*, Cambridge University Press, Cambridge.

Nicholas ASHER and Zhaohui LUO (2012), Formalisation of coercions in lexical semantics, in E. CHEMLA, V. HOMER, and G. WINTERSTEIN, editors, *Proceedings of Sinn und Bedeutung 17*, pp. 63–80, Paris, http://semanticsarchive.net/sub2012/.

David I. BEAVER (2001), *Presupposition and Assertion in Dynamic Semantics*, CSLI Publications, Stanford.

Daisuke BEKKI (2014), Representing anaphora with dependent types, in N. ASHER and S. SOLOVIEV, editors, *Logical Aspects of Computational Linguistics: 8th International Conference, LACL 2014, Proceedings*, volume 8535 of *Lecture Notes in Computer Science*, pp. 14–29, Springer, Heidelberg.

Daisuke BEKKI and Nicholas ASHER (2013), Logical polysemy and subtyping, in Y. MOTOMURA, A. BUTLER, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2012 Workshops, Revised Selected Papers*, volume 7856 of *Lecture Notes in Computer Science*, pp. 17–24, Springer, Heidelberg.

Daisuke BEKKI and Koji MINESHIMA (2017), Context-passing and underspecification in Dependent Type Semantics, in S. CHATZIKYRIAKIDIS and Z. LUO, editors, *Modern Perspectives in Type-Theoretical Semantics*, volume 98 of *Studies in Linguistics and Philosophy*, pp. 11–41, Springer, Heidelberg.

Daisuke BEKKI and Miho SATOH (2015), Calculating projections via type checking, in R. COOPER and C. RETORÉ, editors, *ESSLLI proceedings of the TYTLES workshop on Type Theory and Lexical Semantics ESSLLI2015*, Barcelona.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014), Natural language inference in Coq, *Journal of Logic, Language and Information*, 23(4):441–480.

Robin COOPER (1983), *Quantification and Syntactic Theory*, Reidel, Dordrecht.

Rogelio DÁVILA-PÉREZ (1995), *Semantics and Parsing in Intuitionistic Categorial Grammar*, Ph.D. thesis, University of Essex.

Paul EGRÉ (2008), Question-embedding and factivity, *Grazer Philosophische Studien*, 77(1):85–125.

Delia Graff FARA (2001), Descriptions as predicates, *Philosophical Studies*, 102:1–42, originally published under the name "Delia Graff".

Chris FOX and Shalom LAPPIN (2005), *Foundations of Intensional Semantics*, Blackwell, Oxford.

Jonathan GINZBURG (1995a), Resolving questions, I, *Linguistics and Philosophy*, 18(5):459–527.

Jonathan GINZBURG (1995b), Resolving questions, II, *Linguistics and Philosophy*, 18(6):567–609.

Jeroen GROENENDIJK and Martin STOKHOF (1991), Dynamic predicate logic, *Linguistics and Philosophy*, 14(1):39–100.

Irene HEIM (1982), *The Semantics of Definite and Indefinite Noun Phrases*, Ph.D. thesis, University of Massachusetts, Amherst.

Irene HEIM (1983), On the projection problem for presuppositions, in M. BARLOW, D. FLICKINGER, and M. WESCOAT, editors, *Proceedings of the Second West Coast Conference on Formal Linguistics*, pp. 114–125, Stanford University Press, Stanford, CA.

Jaakko HINTIKKA (1962), *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press, Ithaca, NY.

Jaakko HINTIKKA (1969), Semantics for propositional attitudes, in J. W. DAVIS, D. J. HOCKNEY, and W. K. WILSON, editors, *Philosophical Logic*, volume 20 of *Synthese Library*, pp. 21–45, Reidel, Dordrecht.

William Alvin HOWARD (1980), The formulae-as-types notion of construction, in J. P. SELDIN and J. R. HINDLEY, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 480–490, Academic Press, London.

Hans KAMP, Josef VAN GENABITH, and Uwe REYLE (2011), Discourse Representation Theory, in D. M. GABBAY and F. GUENTHNER, editors, *Handbook of Philosophical Logic*, volume 15, pp. 125–394, Springer, Heidelberg.

Lauri KARTTUNEN (1971), Implicative verbs, *Language*, 47(2):340–358.

Lauri KARTTUNEN and Stanley PETERS (1979), Conventional implicatures, in C. K. OH and D. A. DINNEEN, editors, *Syntax and Semantics 11: Presupposition*, pp. 1–56, Academic Press, New York, NY.

Jeffrey C. KING (2002), Designating propositions, *The Philosophical Review*, 111(3):341–371.

Eriko KINOSHITA, Koji MINESHIMA, and Daisuke BEKKI (2016), An analysis of selectional restrictions with Dependent Type Semantics, in *Proceedings of the 13th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS13)*, pp. 100–113, Kanagawa.

Paul KIPARSKY and Carol KIPARSKY (1970), Fact, in M. BIERWISCH and K. E. HEIDOLPH, editors, *Progress in Linguistics*, pp. 143–173, de Gruyter Mouton, Berlin.

Emiel KRAHMER and Paul PIWEK (1999), Presupposition projection as proof construction, in H. BUNT and R. MUSKENS, editors, *Computing Meaning: Volume 1*, volume 73 of *Studies in Linguistics and Philosophy*, pp. 281–300, Kluwer Academic Publishers, Dordrecht.

Yusuke KUBOTA and Robert LEVINE (2017), Scope parallelism in coordination in Dependent Type Semantics, in M. OTAKE, S. KURAHASHI, Y. OTA, K. SATOH, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2015 Workshops, Revised Selected Papers*, volume 10091 of *Lecture Notes in Artificial Intelligence*, pp. 149–162, Springer, Heidelberg.

Susumu KUNO (1970), Some properties of non-referential noun phrases, in R. JAKOBSON and S. KAWAMOTO, editors, *Studies in General and Oriental Linguistics. Presented to S. Hattori on Occasion of his Sixtieth Birthday*, pp. 348–373, TEC, Tokyo.

Zhaohui LUO (2012a), Common nouns as types, in D. BÉCHET and A. DIKOVSKY, editors, *Logical Aspects of Computational Linguistics: 7th International Conference, LACL 2012, Proceedings*, volume 7351 of *Theoretical Computer Science and General Issues*, pp. 173–185, Springer, Heidelberg.

Zhaohui LUO (2012b), Formal semantics in modern type theories with coercive subtyping, *Linguistics and Philosophy*, 35(6):491–513.

Per MARTIN-LÖF (1984), *Intuitionistic Type Theory. Notes by G. Sambin*, Bibliopolis, Naples.

John-Jules Ch MEYER and Wiebe VAN DER HOEK (2004), *Epistemic Logic for AI and Computer Science*, Cambridge University Press, Cambridge.

Line MIKKELSEN (2005), *Copular Clauses: Specification, Predication and Equation*, John Benjamins, Amsterdam.

Line MIKKELSEN (2011), Copular clauses, in C. MAIENBORN, K. VON HEUSINGER, and P. PORTNER, editors, *Semantics: An International Handbook of Natural Language Meaning*, volume 2, pp. 1805–1829, de Gruyter Mouton, Berlin.

Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in P. SUPPES, J. MORAVCSIK, and J. HINTIKKA, editors, *Approaches to Natural Language*, pp. 221–242, Kluwer Academic Publishers, Dordrecht.

Bengt NORDSTRÖM, Kent PETERSSON, and Jan M. SMITH (1990), *Programming in Martin-Löf's Type Theory: An Introduction*, Oxford University Press, Oxford.

Terence PARSONS (1993), On denoting propositions and facts, *Philosophical Perspectives*, 7:441–460.

Paul PIWEK and Emiel KRAHMER (2000), Presuppositions in context: constructing bridges, in P. BONZON, M. CAVALCANTI, and R. NOSSUM, editors, *Formal Aspects of Context*, volume 20 of *Applied Logic Series*, pp. 85–106, Kluwer Academic Publishers, Dordrecht.

Aarne RANTA (1994), *Type-Theoretical Grammar*, Oxford University Press, Oxford.

Christian RETORÉ (2014), The Montagovian generative lexicon ΛTy_n: a type theoretical framework for natural language semantics, in R. MATTHES and A. SCHUBERT, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics*, pp. 202–229, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, `http://drops.dagstuhl.de/opus/volltexte/2014/4633/`.

Bertrand RUSSELL (1919), *Introduction to Mathematical Philosophy*, George Allen & Unwin, London.

Mark STEEDMAN (2000), *The Syntactic Process*, MIT Press, Cambridge, MA.

Göran SUNDHOLM (1986), Proof Theory and Meaning, in D. M. GABBAY and F. GUENTHNER, editors, *Handbook of Philosophical Logic*, volume 3, pp. 471–506, Reidel, Dordrecht.

Göran SUNDHOLM (1989), Constructive generalized quantifiers, *Synthese*, 79(1):1–12.

Ribeka TANAKA (2014), A proof-theoretic approach to generalized quantifiers in dependent type semantics, in R. DE HAAN, editor, *Proceedings of the ESSLLI2014 Student Session*, pp. 140–151, Tübingen, `http://www.kr.tuwien.ac.at/drm/dehaan/stus2014/proceedings.pdf`.

Ribeka TANAKA, Koji MINESHIMA, and Daisuke BEKKI (2015), Resolving modal anaphora in Dependent Type Semantics, in T. MURATA, K. MINESHIMA, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2014 Workshops, Revised Selected Papers*, pp. 83–98, Springer, Heidelberg.

Ribeka TANAKA, Yuki NAKANO, and Daisuke BEKKI (2014), Constructive generalized quantifiers revisited, in Y. NAKANO, K. SATOH, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2013 Workshops, Revised Selected Papers*, volume 8417 of *Lecture Notes in Computer Science*, pp. 115–124, Springer, Heidelberg.

Wataru UEGAKI (2016), Content nouns and the semantics of question-embedding, *Journal of Semantics*, 33(4):623–660.

Rob A. VAN DER SANDT (1992), Presupposition projection as anaphora resolution, *Journal of Semantics*, 9:333–377.

Zeno VENDLER (1972), *Res Cogitans: An Essay in Rational Psychology*, Cornell University Press, Ithaca, NY.