Questions in Dependent Type Semantics

Kazuki Watanabe
The University of Tokyo
watanabe-kazuki163@g.ecc.u-tokyo.ac.jp

Koji Mineshima
Ochanomizu University
mineshima.koji@ocha.ac.jp

Daisuke Bekki
Ochanomizu University
bekki@is.ocha.ac.jp

1 Introduction

Dependent Type Semantics (DTS; Bekki and Mineshima (2017)) is a semantic framework that provides a unified analysis of presupposition and anaphora, based on dependent type theory (Martin-Löf, 1984). The semantic representations for declarative sentences in DTS are *types*, based on the propositions-as-types paradigm. While type-theoretic semantics for natural language based on dependent type theory has been developed (Ranta, 1994; Luo, 2012; Cooper, 2012; Chatzikyriakidis and Luo, 2017, among others), how to assign semantic representations to *interrogative* sentences in such a framework has been a non-trivial problem. In this study, we show how to provide the semantics of interrogative sentences in DTS. The basic idea is to assign the same type to both declarative sentences and interrogative sentences, partly building on the recent proposal in Inquisitive Semantics (Ciardelli et al., 2019), where interrogative and declarative sentences are treated as having the same type.

While our extension of DTS adopts some notions from Inquisitive Semantics, there is a difference between the two approaches. Crucially, while double negation is a key to make a distinction between assertion and question in Inquisitive Semantics, we do not make use of double negation for this purpose because it blocks anaphoric links in terms of Σ-types in DTS (see section 3.4 for the detail).

Another difference is that DTS is based on the idea of proof-theoretic semantics where the meaning of a sentence is given in terms of inference rules. The proof-theoretic approach is particularly suited for computational approaches to semantics and natural language inference; we use Combinatory Categorial Grammar (CCG; Steedman (1996)) as a syntactic component of DTS and implement our compositional semantics for interrogative sentences using ccg2lambda[1] (Martínez-Gómez et al., 2016), a semantic parsing platform based on CCG. Also, on the basis of the idea that the relationship between a question and an answer can be formulated as a task of Recognizing Textual Entailment (RTE), we implement our inference system using proof assistant Coq (The Coq Development Team, 2016)[2] and show that our system can deal with a wide range of question-answer relationships.[3] For this purpose, we build a testset to evaluate interrogative semantics and inference system, which consists of 49 question-answer pairs discussed in the formal semantics literature. Using proof automation in Coq, we implement a proof system for DTS that can prove these question-answer relationships formulated as RTE problems.

In short, the contributions of this research are threefold: (i) to present a compositional semantics in DTS for various types of questions, including polar questions, alternative questions, and wh-questions; (ii) to implement our compositional semantics and proof system to solve question answering as RTE problem; and (iii) to create a testset compiling question-answer pairs discussed in the literature and evaluate our implemented system on it.

[1] https://github.com/mynlp/ccg2lambda

[2] See Chatzikyriakidis and Luo (2014) for the use of Coq in formalizing natural language inferences in dependent type theory.

[3] The system will be available at https://github.com/Kazuuuuuki/DTS-question-parser.

$$\frac{\begin{array}{c}\overline{x:A}\ \mathrm{k}\\ \vdots\\ A:type_i \quad B:type_j\end{array}}{(x:A)\to B:type_{max(i,j)}}\ (\Pi F),\mathrm{k} \qquad \frac{\begin{array}{c}\overline{x:A}\ \mathrm{k}\\ \vdots\\ A:type_i \quad M:B\end{array}}{\lambda x.M:(x:A)\to B}\ (\Pi I),\mathrm{k} \qquad \frac{M:(x:A)\to B \quad N:A}{MN:B[N/x]}\ (\Pi E)$$

$$\frac{\begin{array}{c}\overline{x:A}\ \mathrm{k}\\ \vdots\\ A:type_i \quad B:type_j\end{array}}{\left[\begin{array}{l}x:A\\ B\end{array}\right]:type_{max(i,j)}}\ (\Sigma F),\mathrm{k} \qquad \frac{M:A \quad N:B[M/x]}{(M,N):\left[\begin{array}{l}x:A\\ B\end{array}\right]}\ (\Sigma I) \qquad \frac{M:\left[\begin{array}{l}x:A\\ B\end{array}\right]}{\pi_1(M):A}\ (\Sigma E) \qquad \frac{M:\left[\begin{array}{l}x:A\\ B\end{array}\right]}{\pi_2(M):B[\pi_1(M)/x]}\ (\Sigma E)$$

$$\frac{A:type_i \quad B:type_j}{A \biguplus B:type_{max(i,j)}}\ (\biguplus F) \qquad \frac{M:A}{\iota_1(M):A\biguplus B}\ (\biguplus I) \qquad \frac{M:B}{\iota_2(M):A\biguplus B}\ (\biguplus I)$$

$$\frac{\begin{array}{cccc}L:A\biguplus B & C:(A\biguplus B)\to type_i & \begin{array}{c}\overline{x:A}\ \mathrm{k}\\ \vdots\\ M:C(\iota_1(x))\end{array} & \begin{array}{c}\overline{x:B}\ \mathrm{k}\\ \vdots\\ N:C(\iota_2(x))\end{array}\end{array}}{case\ L\ of\ (\lambda x.M;\lambda x.N):C(L)}\ (\biguplus E),k$$

$$\frac{}{\{a_1,\ldots,a_n\}:type_i}\ (\{\}\ \mathrm{F}) \qquad \frac{}{a_k:\{a_1,\ldots,a_n\}}\ (\{\}\ \mathrm{I})$$

$$\frac{M:\{a_1,\ldots,a_n\} \quad C:\{a_1,\ldots,a_n\}\to type_i \quad N_1:C(a_1) \quad \ldots \quad N_n:C(a_n)}{case_M(N_1,\ldots,N_n):C(M)}\ \{\}\ \mathrm{E}$$

Figure 1: Inference rules

There exist other semantic frameworks based on dependent type theory which deal with interrogative sentences. Ginzburg (2005) assigns different types to declarative sentences (assertion) and interrogative sentences (question); assertion is assigned a record type, while question is assigned the type of functions that maps records to propositions. In Ranta (1994), some meta-rules are introduced for describing the relationship of answers to the corresponding questions. A detailed comparison between our framework and these other type-theoretic frameworks must be left for another occasion.

The structure of the paper is as follows. In section 2 we briefly introduce the framework of DTS (for more detail, see Bekki (2014); Bekki and Mineshima (2017); Tanaka et al. (2018)). The main part of this paper is section 3. In this section, we extend the basic theory of DTS and present semantic representations for basic interrogative sentences. We show that this extension preserves the analysis of anaphora in terms of Σ-types in DTS. In section 4, we give an overview of how to provide a compositional semantics to derive semantic representations using CCG as a syntactic framework. In section 5, we present a question-answering testset for evaluating interrogative semantics and the evaluation of our system on the testset. In section 6, we briefly discuss some future work.

2 DTS

In this section, we explain a basic framework of DTS that is relevant to our proposal in this paper. As mentioned in section 1, a proposition (which corresponds to a semantic representation of a sentence) is regarded as a *type* in DTS. In our analysis, we use the following four type constructors (in the DTS notation) which are also used in the previous study on formal semantics based on dependent type theory (Ranta, 1994; Luo, 2012; Bekki and Mineshima, 2017).

- Π-type: $(x{:}\,A)\to B(x)$
- Σ-type: $(x:A)\times B$, also written as $\left[\begin{array}{l}x:A\\ B(x)\end{array}\right]$
- Disjoint Union Type ($\biguplus$-type): $A\biguplus B$

$$\frac{\begin{array}{c}\overline{x:A}\ \mathrm{k}\\ \vdots\\ A:type_i \qquad B:type_j\end{array}}{(x{:}\,A)\bigoplus B:type_{max(i,j)}}\ (\bigoplus F),\mathrm{k} \qquad \frac{t:A \qquad u:B[t/x]}{[t,u]:(x{:}\,A)\bigoplus B}\ (\bigoplus I) \qquad \frac{\begin{array}{c}\overline{x:A}\ \mathrm{k}\ \ \overline{y:B(x)}\ \mathrm{k}\\ \vdots\\ [t,u]:(x{:}\,A)\bigoplus B \qquad m:C\end{array}}{case_{[t,u]}m:C}\ (\bigoplus E),\mathrm{k}$$

Figure 2: Inference rules of existential type. Elimination rule $(\bigoplus E)$ can be applied if $m : C$ and any open assumption on which $m : C$ depends do not contain x and y as free variables.

- Enumeration Type ($\{\ \}$-type): $\{a_1, a_2, \ldots, a_n\}$

Figure 1 shows inference rules for these types. Π-type and Σ-type can be considered as generalized function type and generalized product type, respectively. These two types make a difference from simple type theory where only non-dependent function type $A \to B$ and product type $A \times B$ are admitted. $\biguplus$-type is a disjoint union type. We also use enumeration type ($\{\ \}$-type), written $\{a_1, a_2, \ldots, a_n\}$, to express the finite domain of entities; this setting is essential for our implementation, which we will discuss in section 5. The bottom type $\bot$ is defined as $\{\ \}$, i.e., the enumeration type inhabited by no term. The bottom type $\bot$ is used for defining the negation of A; as usual, $\neg A$ is defined as $A \to \bot$.

In addition, we use *existential type* (also called weak-sigma type) (Luo, 1994), written $(x{:}\,A)\bigoplus B$, for semantic representations of wh-questions. Figure 2 shows inference rules of $\bigoplus$-type. Existential type corresponds to existential quantification in intuitionistic logic. The difference between Σ-type and $\bigoplus$-type is in elimination rule. The elimination rule of Σ-type allows to use projections (see ΣE in Figure 1), while the elimination rule of $\bigoplus$-type (see $\bigoplus E$ in Figure 2) does not. Thus, a pair of terms $[t, u]$ which is a proof term of $(x{:}\,A)\bigoplus B$ cannot be divided into t and u by applying projection.

One of the other features of DTS is that expressions that trigger presupposition and anaphora are uniformly treated as underspecified terms, written @ (See Bekki and Mineshima (2017) for the detail). In our extension of DTS with interrogative semantics, this uniform treatment of anaphora and presupposition in terms of underspecification is preserved.

3 Semantic Representations for Interrogatives

In this section, we introduce semantic representations of interrogative sentences. We focus on three types of questions: polar question, alternative question and wh-question. Before we explain the detail of semantics of interrogative sentences in DTS, we show how to characterize the relationships between questions and their answers in our framework.

Partly building on Inquisitive Semantics (Ciardelli et al., 2019), we treat questions and assertions as having the same type in our type-theoretic framework. Also following Inquisitive Semantics, we define the entailment relation holding between a semantic representation (SR) of a declarative sentence and that of an interrogative sentence: the SR S_1 of a declarative sentence is an answer to the SR S_2 of an interrogative sentence if and only if S_1 entails S_2 in DTS. Using this definition, we can describe question-answer relationship as entailment relation and evaluate our semantic representations by a testset for question-answering.

3.1 Basic Declarative Sentence

We start with semantic representations of basic declarative sentences in DTS. Σ-type represents existential sentences; for instance, (1b) is a semantic representation of (1a). Σ-type is used to capture *externally dynamic* character of existential quantification in the sense of Groenendijk and Stokhof (1991); we will come back to this point later.

(1) a. Someone ran.

b. $\left[\begin{array}{l} x : \text{Entity} \\ \textbf{run}(x) \end{array}\right]$

Another type of basic declarative sentence we need to introduce is a universal sentence like (2a). The semantic representation of (2a) is given in (2b). Universal propositions are analyzed using Π-type.

(2) a. Every student ran.

b. $\left(u\colon \left[\begin{array}{l} x : \text{Entity} \\ \textbf{student}(x) \end{array}\right]\right) \rightarrow \textbf{run}(\pi_1(u))$

3.2 Wh-Question

There are two different interpretations of wh-question, called *mention-some* reading and *mention-all* reading (Dayal, 2016). For instance, (3a) and (3b) are examples of wh-questions having a mention-some reading and a mention-all reading, respectively.

(3) a. What is something that Alice really likes?

b. Who did Alice invite to her birthday party?

An answer to a mention-some wh-question is characterized by an instance satisfying the property in question; thus it does not have to mention all instances which satisfy the property. For example, if Alice likes chocolate, football and mathematics, (4a) and (4b) can be regarded as an answer to the mention-some wh-question in (3a), that is, an answer to (3a) is characterized by mentioning some entity which Alice likes.

(4) a. Alice likes chocolate.

b. Alice likes football.

An answer to a mention-all question has to be exhaustive, that is, it must provide complete information about the question in the relevant domain. Thus, if Alice invited only Susan and John to her birthday party, (5a) is an answer to (3b). By contrast, (5b) and (5c) are not suited for an answer to (3b) because they do not give an exhaustive answer.

(5) a. Alice invited only Susan and John to her birthday party.

b. Alice invited Susan.

c. Alice invited John.

For representing the meaning of a mention-some wh-question, we use existential type ($\bigoplus$-type). For the sake of illustration, consider a simple mention-some wh-question in (6a), whose semantic representation is given in (6b).

(6) a. Who ran? (mention-some reading)

b. $(x\colon \text{Entity}) \bigoplus \textbf{run}(x)$

The existential sentence in (1a) can be a (at least semantically) proper answer to the mention-some wh-question. Thus the entailment relation in (7a) should hold. What is crucial here is that declarative existential sentences are analyzed as Σ-types, while mention-some wh-questions are analyzed as existential types. Thus, our analysis correctly derives the relation as stated in (7b); the SR of (1a) entails the SR of (6a), but not vice versa.[4]

(7) a. Someone ran. $\Rightarrow$ Who ran? (mention-some)

b. $\left[\begin{array}{l} x : \text{Entity} \\ \textbf{run}(x) \end{array}\right] \vdash (x\colon \text{Entity}) \bigoplus \textbf{run}(x)$

As shown in Figure 1 and Figure 2, Σ-type and $\bigoplus$-type have the same formation and introduction rules. As is mentioned in Section 2, the difference is in elimination rule. This causes differences in anaphora resolution between (8a) and (8b).

[4]It is widely accepted that mention-some wh-question has an existential presupposition (Dayal, 2016). Here we assume that a mention-some wh-question does not entail the corresponding existential sentence but just presupposes it.

(8) a. Someone$_i$ ran. He$_i$ is a student.
b. Who$_i$ ran? (mention-some) # He$_i$ is a student.

An existential proposition introduces an entity which can be referred to in the subsequent sentences (Groenendijk and Stokhof, 1991). This externally dynamic character of existential quantification is captured by means of Σ-types (Ranta, 1994; Bekki and Mineshima, 2017; Tanaka et al., 2018). Thus, the mini-discourse in (8a) can be given the following full interpretation in DTS.[5]

(9) $$\left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{run}(x) \end{array}\right] \\ \mathbf{student}(\pi_1(u)) \end{array}\right]$$

Here, u introduced by the Σ-type is a pair of an entity x and a proof that x is a student. Thus the projection $\pi_1(u)$, which is allowed by the elimination rule of Σ-type (see Figure 1), successfully picks up its first component (the entity x), which can be used in the subsequent discourse.

In contrast, the elimination rule of $\bigoplus$-type (see Figure 2) does not provide a projection function. Thus this makes it impossible to establish the anaphoric link for (8b), which is a desirable prediction. There is no way to pass an entity introduced by the SR in (10) to the subsequent sentences.

(10) $(x\colon \text{Entity}) \bigoplus \mathbf{run}(x)$

Let us move on to the semantic representations of mention-all wh-question in our framework. For instance, consider the question in (11a). The mention-all reading of this question can be represented using Π-type and disjoint union type as in (11b).

(11) a. Who ran? (mention-all)
b. $(x\colon \text{Entity}) \to (\mathbf{run}(x) \uplus \neg\mathbf{run}(x))$

As is in other systems based on dependent types, our underlying proof system is based on intuitionistic logic where the law of excluded middle is not allowed. Therefore, (11b) is not a theorem. A proof for the SR in (11b) is a function f such that for any entity x, $f(x)$ is a proof of $\mathbf{run}(x)$ or a proof of $\neg\mathbf{run}(x)$. That is to say, to prove the SR in (11b), one has to know whether x runs or not for each entity x in the domain. This naturally captures the answering condition for the mention-all reading of (11a). Note that (12b) can serve as an answer to this mention-all question.

(12) a. Only John ran.
b. $((x\colon \text{Entity}) \to (\mathbf{run}(x) \to (x = \mathbf{j}))) \wedge \mathbf{run}(\mathbf{j})$

The SR in (12b) means that John ran and the other entities did not run. We may assume that the number of entities in the domain is finite, which can be expressed by using enumeration type ($\{\}$-type). In this setting, (12b) entails (11b) in DTS; thus this correctly predicts that (12a) is an answer to (11a).

3.3 Polar Question

Semantic representations of polar question are given by using $\uplus$ -type. (13a) is a simple example of polar question and (13b) is its semantic representation.

(13) a. Did John run?
b. $\mathbf{run}(\mathbf{j}) \uplus \neg\mathbf{run}(\mathbf{j})$

In the same manner as in (11), our analysis derives the entailment in (14b), thus correctly predicting that *John ran* is an answer to the polar question in (13a).

(14) a. John ran. $\Rightarrow$ Did John run?
b. $\mathbf{run}(\mathbf{j}) \vdash \mathbf{run}(\mathbf{j}) \uplus \neg\mathbf{run}(\mathbf{j})$

[5]For the detail on how to derive this interpretation using underspecified terms in DTS, see Tanaka et al. (2018).

3.4 Alternative Question

The semantic representations of alternative questions are also given by using $\biguplus$-type. While the semantic representation of a polar question automatically meets the exhaustiveness condition by using $\biguplus$-type, some alternative questions need to express exhaustiveness explicitly. Here we assume that *neither* and *both* are not a suitable answer to an alternative question like (15a).[6] We use (15b) as the semantic representation of (15a).

(15) a. Did John run or walk ?

b. $\mathbf{run}(\mathbf{j}) \biguplus \mathbf{walk}(\mathbf{j})$
$\wedge\, (\mathbf{run}(\mathbf{j}) \rightarrow (\neg \mathbf{walk}(\mathbf{j})))$
$\wedge\, (\mathbf{walk}(\mathbf{j}) \rightarrow (\neg \mathbf{run}(\mathbf{j})))$

Under this analysis, it can be shown that (16a), whose semantic representation is given in (16b), is an answer to (15a).

(16) a. John ran and didn't walk.

b. $\mathbf{run}(\mathbf{j}) \wedge \neg \mathbf{walk}(\mathbf{j})$

In Inquisitive Semantics, alternative questions and declarative sentences with *or* are logically distinguished by means of double negation (Ciardelli et al., 2019). This option is not available in our framework, because double negation wrongly blocks a certain type of potential anaphoric links in DTS. As an example, consider the mini-discourse in (17a). The semantic representation of the first sentence of (17a) is given in (17b).

(17) a. Susan$_i$ saw a horse$_j$ or a pony$_j$. She$_i$waved to it$_j$.

b. $$\left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{horse}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right] \biguplus \left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{pony}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right]$$
$$\wedge \left(\left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{horse}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right] \rightarrow \neg \left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{pony}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right]\right)$$
$$\wedge \left(\left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{pony}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right] \rightarrow \neg \left[\begin{array}{l} u : \left[\begin{array}{l} x : \text{Entity} \\ \mathbf{horse}(x) \end{array}\right] \\ \mathbf{see}(\mathbf{s}, \pi_1 u) \end{array}\right]\right)$$

If the semantic representation of the first sentence of (17a) is the one obtained by applying double negation to (17b), then it predicts that the anaphoric link in (17a) is impossible, contrary to the fact. For this reason, we do not use double negation for distinguishing alternative question from declarative disjunctive sentence.

4 Compositional Semantics

In this section, we provide a brief overview of how to compose semantic representations of sentences in DTS. For concreteness, we use CCG as a syntactic component of our framework. Table 1 shows an excerpt from the lexical entries we implement in this study.[7] For convenience, we assume that the category NP is always type-raised; the type-raised categories $(S/(S\backslash NP))$ and $(S\backslash(S/NP))$ are abbreviated as $NP^{\uparrow}_{nom}$ and $NP^{\uparrow}_{acc}$, respectively. (18) is a simple example of the derivation tree annotated with semantic representations.

[6] See Biezma and Rawlins (2012) for discussion on the status of "neither" and "both" as an answer to alternative questions.

[7] All the codes to implement our system and the testset used in the experiments are available at `https://github.com/Kazuuuuuki/DTS-question-parser`.

PF	Category	Semantic representation
John	$NP^{\uparrow}_{nom\|acc}$	$\lambda p.p(\mathbf{j})$
	$S_{or}/(S_{or}\backslash NP)$	$\lambda p.p(\mathbf{j})$
everyone	$NP^{\uparrow}_{nom\|acc}$	$\lambda p.\,(x{:}\,\text{Entity}) \to p(x)$
someone	$NP^{\uparrow}_{nom\|acc}$	$\lambda p.\begin{bmatrix} x : \text{Entity} \\ p(x) \end{bmatrix}$
nobody	$NP^{\uparrow}_{nom\|acc}$	$\lambda p.\,(x{:}\,\text{Entity}) \to \neg p(x)$
every	$NP^{\uparrow}_{nom\|acc}/N$	$\lambda n.\lambda p.\left(v{:}\begin{bmatrix} y : \text{Entity} \\ ny \end{bmatrix}\right) \to p(\pi_1(v))$
a, some	$NP^{\uparrow}_{nom\|acc}/N$	$\lambda n.\lambda p.\begin{bmatrix} v : \begin{bmatrix} y : \text{Entity} \\ ny \end{bmatrix} \\ p(\pi_1(v)) \end{bmatrix}$
only	$NP^{\uparrow}_{nom}/NP^{\uparrow}_{nom}$	$\lambda P.\lambda q.((x{:}\,\text{Entity}) \to (q(x) \to P(\lambda y.y) = x)) \wedge P(q)$
	$NP^{\uparrow}_{acc}/NP^{\uparrow}_{acc}$	$\lambda P.\lambda q.((x{:}\,\text{Entity}) \to (q(x) \to P(\lambda y.y) = x)) \wedge P(q)$
student	N	**student**
walk	$S\backslash NP$	**walk**
run	$S\backslash NP$	**run**
see	$S\backslash NP^{\uparrow}_{nom}/NP^{\uparrow}_{acc}$	$\lambda Q.\lambda P.P(\lambda x.Q(\lambda y.\mathbf{see}(x,y)))$
or	$(S\backslash S)/S$	$\lambda S1.\lambda S2.S1 \biguplus S2$
	$(S_{or}\backslash S)/S$	$\lambda S1.\lambda S2.((S1 \biguplus S2) \wedge (S1 \to \neg S2) \wedge (S2 \to \neg S1))$
	$((S\backslash NP)\backslash(S\backslash NP))/(S\backslash NP)$	$\lambda p.\lambda q.\lambda x.(q(x) \biguplus p(x))$
	$((S_{or}\backslash NP)\backslash(S\backslash NP))/(S\backslash NP)$	$\lambda p.\lambda q.\lambda x.(q(x) \biguplus p(x)) \wedge (q(x) \to \neg p(x)) \wedge (p(x) \to \neg q(x))$
and	$(S\backslash S)/S$	$\lambda S1.\lambda S2.S1 \wedge S2$
	$(NP^{\uparrow}_{nom}\backslash NP^{\uparrow}_{nom})/NP^{\uparrow}_{nom}$	$\lambda P.\lambda Q.\lambda r.P(r) \wedge Q(r)$
	$(NP^{\uparrow}_{acc}\backslash NP^{\uparrow}_{acc})/NP^{\uparrow}_{acc}$	$\lambda P.\lambda Q.\lambda r.P(r) \wedge Q(r)$
who	$S_{some}/(S\backslash NP)$	$\lambda p.\,(x{:}\,\text{Entity}) \bigoplus p(x)$
	$S_{all}/(S\backslash NP)$	$\lambda p.\,(x{:}\,\text{Entity}) \to p(x) \biguplus \neg p(x)$
	$S_{some}/((S\backslash NP^{\uparrow}_{nom})/NP^{\uparrow}_{acc})/NP^{\uparrow}_{nom}$	$\lambda P.\lambda Q.Q(\lambda Q1.\begin{bmatrix} x : \text{Entity} \\ Q1(x) \end{bmatrix}(P))$
do	S/S	$\lambda S.S$
	S_{or}/S_{or}	$\lambda S.S$
did	S/S	$\lambda S.S$
	S_{or}/S_{or}	$\lambda S.S$
?	$S\backslash S$	$\lambda S.(S \biguplus \neg S)$
	$S\backslash S_{some}$	$\lambda S.S$
	$S\backslash S_{or}$	$\lambda S.S$
	$S\backslash S_{all}$	$\lambda S.S$

Table 1: An excerpt from the lexical entries for our interrogative semantics.

(18) John saw Susan.

$$\dfrac{\dfrac{\text{John}}{NP^{\uparrow}_{nom}}\ \lambda p.p(\mathbf{j}) \qquad \dfrac{\dfrac{\text{saw}}{(S\backslash NP^{\uparrow}_{nom})/NP^{\uparrow}_{acc}}\ \lambda Q.\lambda P.P(\lambda x.Q(\lambda y.\mathbf{see}(x,y))) \qquad \dfrac{\text{Susan}}{NP^{\uparrow}_{acc}}\ \lambda p.p(\mathbf{s})}{S\backslash NP^{\uparrow}_{nom}}\ >\ \lambda P.P(\lambda x.\mathbf{see}(x,\mathbf{s}))}{S}\ <\ \mathbf{see}(\mathbf{j},\mathbf{s})$$

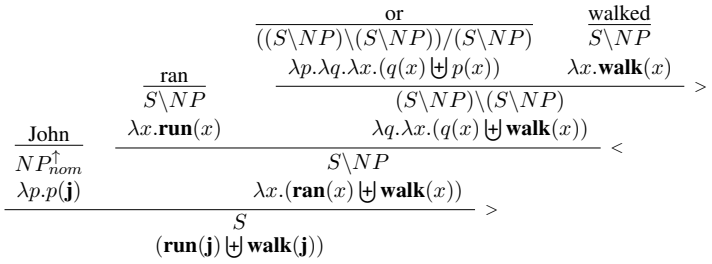
As an example of a simple polar question, consider (19). The CCG derivation of (19) shows that the question mark "?" induces inquisitive meaning for polar questions.

(19) Did John run?

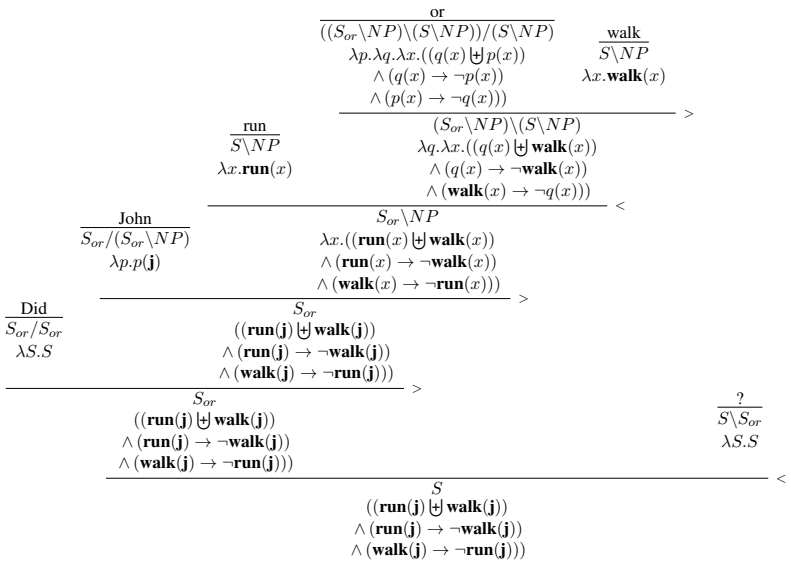
$$\dfrac{\dfrac{\dfrac{\text{Did}}{S/S}\ \lambda S.S \qquad \dfrac{\dfrac{\text{John}}{NP^{\uparrow}_{nom}}\ \lambda p.p(\mathbf{j}) \qquad \dfrac{\text{run}}{S\backslash NP}\ \lambda x.\mathbf{run}(x)}{S}\ >\ \mathbf{run}(\mathbf{j})}{S}\ >\ \mathbf{run}(\mathbf{j}) \qquad \dfrac{\dfrac{?}{S\backslash S}\ \lambda S.(S \biguplus \neg S)}{S}\ <\ (\mathbf{run}(\mathbf{j}) \biguplus \neg \mathbf{run}(\mathbf{j}))$$

As a more involved case, (20) is a derivation tree for a declarative sentence with *or* and (21) for an alternative question with *or*. Note that we use different categories for *or* in (20) and (21).

(20) John ran or walked.



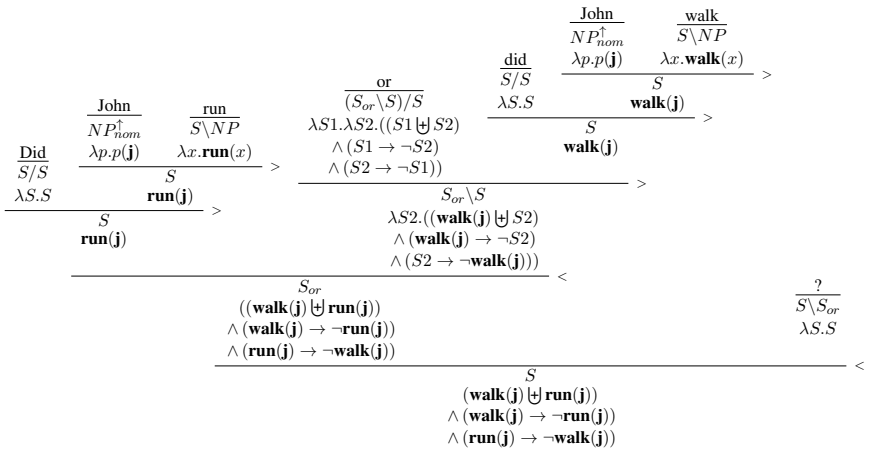
(21) Did John run or walk ?



In (21), *or* introduces a category with a feature S_{or}, which is taken over to the node of category S_{or} for *Did John run or walk*; then the question mark "?" of category $S\backslash S_{or}$ takes it as argument and returns an expression of category S. Thus the question mark "?" does not introduce inquisitive meaning if it takes an expression of category S_{or}.

The reason for assigning several different categories to a question mark can be seen by considering (22).[8] The semantic representation of (22) is one for an alternative question in the same way as (21). Thus the inquisitive meaning is introduced not by the question mark "?" but by *or* of alternative question.

(22) Did John run or did he walk ?



[8] In the derivation tree of (22), *he* is converted to *John* for simplicity.

Answer (Premise)	Question (Conclusion)	Label
John ran.	Who ran ? (mention-some)	yes
Someone ran.	Who ran ? (mention-some)	yes
Only John saw Susan.	Who saw Susan ? (mention-all)	yes
John ran or Susan walked.	Who ran ? (mention-some)	unknown
Nobody ran.	Who ran ? (mention-some)	no
Nobody ran.	Who ran ? (mention-all)	yes
Only John ran.	Who didn't run ? (mention-all)	yes
John didn't run.	Who ran ? (mention-some)	unknown
Every student ran.	Did John run ?	yes
John ran or Susan walked.	Did John run ?	unknown
John ran and Susan walked.	Did John run ?	yes
John didn't run.	Did John run ?	yes
Only John ran.	Did John run ?	yes
John ran or walked.	Did John run or walk?	unknown
John ran and didn't walk.	Did John run or did John walk ?	yes
Nobody ran.	Did John run?	yes
John is tall and John is not short.	Is John tall or short ?	yes

Table 2: Some examples from our testset. We assume that the denotation of *student* consists of John and Susan.

5 Experiment

We implement the compositional semantics introduced in the previous section using ccg2lambda. Since there is a non-trivial gap between the outputs of the state-of-the-art CCG parsers (e.g., Yoshikawa et al., 2017) and the syntactic structures we assume in this study, we manually annotate CCG trees for the input sentences. We implement the compositional mapping from CCG trees to the semantic representations discussed so far, using the lexical entries presented in the previous section.[9] The system automatically converts an input CCG tree to the corresponding semantic representation in DTS.

Implementation of a proof system is also needed for checking the relationships between answers and questions formulated as entailment relation. We use Ltac (Delahaye, 2000), a tactic language available in Coq to implement proof automation necessary for our purposes.

We built a testset consisting of 49 question-answer pairs. Some examples are shown in Table 2. Each problem in the testset has two sentences; the first sentence is an answer and the second sentence is a question. For each pair of the testset, we annotated a gold CCG tree. Here is an example for an answer sentence *Everyone runs* and a question sentence *Who ran ?* (mention-some reading):

(23)
```
(S (<S/<S\NP>> Everyone) (S\NP run))
(S (S_some (<S_some/<S\NP>> Who_some) (S\NP run)) (S\S_some ?))
```

The answer to each problem is *yes* (entailment), *no* (contradiction), or *unknown* (neither), following the FraCaS testset (Cooper et al., 1994). The distribution of answers is: yes/no/unknown = 36/1/12 For evaluating the basic capacity of compositional semantics and proof system, we keep the sentences included in the testset as simple as possible. This makes easy to detect what phenomena a given system fails to give an appropriate semantic representation.

In the setting for theorem proving in Coq, we use a finite domain with three entities: John, Susan and Lucy. This can be implemented using enumeration type in Coq:

(24)
```
Inductive Entity : Type := | John | Susan | Lucy.
```

[9]The distinction between *mention-some* and *mentione-all* readings is annotated to wh-expressions; see entries for *who* in Table 1.

The Coq script automatically generated for the CCG trees for the question-answer pair in (23) is the following:

(25) `(forall x:Entity,(_run x)) -> (ex Entity (fun x => (_run x)))`

Here the semantic representation for the answer sentence appears in the antecedent of implication "`->`" and that for the question sentence appears in the consequent. We use existential type in Coq, written `ex`, for representing mention-some wh-question. The system correctly proves (25) as a theorem.

In this setting, we can successfully derive the desired semantic representations and prove the entailment relations for all 49 cases.

6 Conclusion

In this paper, we extended the framework of DTS with semantics for a variety of interrogative sentences. There are many topics which cannot be discussed in this paper. Among others, we do not deal with embedded questions. Also, we do not give a detailed examination of the presuppositions of wh-questions in the context of DTS. Tanaka et al. (2017) presented an analysis of factive verbs like *know* in the framekwork of DTS, but the relationship between our proposal and this work is not obvious yet. These issues will be left for future work.

Acknowledgement

This work was partially supported by JSPS KAKENHI Grant Number JP18H03284 Japan, and JST CREST Grant Number JPMJCR1301 Japan. We thank the three anonymous reviewers for their helpful comments.

References

Bekki, D. (2014). Representing anaphora with dependent types. In *Logical Aspects of Computational Linguistics (8th international conference)*, pp. 14–29. Springer.

Bekki, D. and K. Mineshima (2017). Context-passing and underspecification in Dependent Type Semantics. In *Modern Perspectives in Type Theoretical Semantics, Studies in Linguistics and Philosophy*, pp. 11–41. Springer.

Biezma, M. and K. Rawlins (2012). Responding to alternative and polar questions. *Linguistics and Philosophy 35*(5), 361–406.

Chatzikyriakidis, S. and Z. Luo (2014). Natural language inference in Coq. *Journal of Logic, Language and Information 23*(4), 441–480.

Chatzikyriakidis, S. and Z. Luo (2017). *Modern Perspectives in Type-Theoretical Semantics*. Springer.

Ciardelli, I., J. Groenendijk, and F. Roelofsen (2019). *Inquisitive semantics*. Oxford University Press.

Cooper, R. (2012). Type theory and semantics in flux. *Handbook of the Philosophy of Science 14*, 271–323.

Cooper, R., R. Crouch, J. Van Eijck, C. Fox, J. Van Genabith, J. Jaspers, H. Kamp, M. Pinkal, M. Poesio, S. Pulman, et al. (1994). FraCaS: A framework for computational semantics. *Deliverable D6*.

Dayal, V. (2016). *Questions*. Oxford University Press.

Delahaye, D. (2000). A tactic language for the system coq. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pp. 85–95. Springer.

Ginzburg, J. (2005). Abstraction and ontology: Questions as propositional abstracts in type theory with records. *Journal of Logic and Computation 15*, 113–130.

Groenendijk, J. and M. Stokhof (1991). Dynamic predicate logic. *Linguistics and philosophy 14*(1), 39–100.

Luo, Z. (1994). *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press.

Luo, Z. (2012). Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy 35*(6), 491–513.

Martin-Löf, P. (1984). *Intuitionistic Type Theory*. Bibliopolis.

Martínez-Gómez, P., K. Mineshima, Y. Miyao, and D. Bekki (2016). ccg2lambda: A compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, pp. 85–90. Association for Computational Linguistics.

Ranta, A. (1994). *Type-Theoretical Grammar*. Oxford University Press.

Steedman, M. (1996). *Surface Structure and Interpretation*. MIT Press.

Tanaka, R., K. Mineshima, and D. Bekki (2017). Factivity and presupposition in dependent type semantics. *Journal of Language Modelling 5*(2), 385–420.

Tanaka, R., K. Mineshima, and D. Bekki (2018). Paychecks, presupposition, and dependent types. In *Fifth Workshop on Natural Language and Computer Science (NLCS 2018)*.

The Coq Development Team (2016). The Coq Proof Assistant: Reference Manual: Version 8.5.pl3.

Yoshikawa, M., H. Noji, and Y. Matsumoto (2017). A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 277–287. Association for Computational Linguistics.