# Towards Modeling Natural Language Inferences with Part-Whole Relations using Formal Ontology and Lexical Semantics

**Ayako Nakamura**
Ochanomizu University
2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan

**Koji Mineshima**
Ochanomizu University
CREST, Japan Science and Technology Agency

**Daisuke Bekki**
Ochanomizu University
CREST, Japan Science and Technology Agency

**Abstract**

In this paper, we present a framework of natural language semantics combined with formal ontology to deal with lexical and world knowledge. We build on a framework of Dependent Type Semantics (DTS), a framework of natural language semantics based on dependent type theory. We show how to handle natural language inferences with part-whole relations, in particular, bridging inferences and inferences with the so-called total and partial predicates, in this framework.

## Introduction

Entailment relations are of central importance in the study of formal semantics for natural languages. Generally speaking, the task of determining whether one sentence intuitively entails another sentence requires vast amounts of lexical and world knowledge. Over the past several decades, however, formal semanticists have concentrated on a relatively small set of entailment relations that arise from the compositional structure of a sentence, and in doing so, have abstracted away from how logical inferences interact with a rich body of lexical and world knowledge.

Meanwhile, since the emergence of statistical parsers based on sophisticated syntactic theories (Clark and Curran 2007), there has been developed a wide-coverage semantic parser that translates sentences into logical formulas and recognizes entailments using theorem proving (Bos 2008). Then it has been of increasing importance to combine well-developed methods of formal semantics with a rich body of lexical and world knowledge for natural language inferences. For that purpose, large lexical resources such as WordNet (Fellbaum 1998) have been widely used, and there have been attempts to improve the quality of such resources using the concepts of formal ontology (Gangemi et al. 2003). At present, however, there are few attempts to combine such an ontology with the state-of-the-art formal semantics; moreover, there is little discussion on what kind of knowledge is needed to represent a variety of inferences in natural languages from a linguistic point of view.

In this paper, we propose a framework of natural language semantics combined with formal ontology to deal with lexical and world knowledge. We build on a framework of Dependent Type Semantics (DTS), a framework of natural language semantics based on dependent type theory (Martin-Löf 1984). A special attention will be paid to the inferences that are sensitive to the part-whole relations among structured entities, in particular, bridging inferences and inferences with total and partial predicates.

Formal semantics has been developed as sentence semantics and then further extended to discourse semantics in 1980's. However, the attempt to combine formal semantics with lexical semantics is still underdeveloped. The present paper also aims to contribute to filling this gap, by means of enriching type-theoretical semantics of DTS with a mechanism to handle inferences based on lexical knowledge.



## Textual entailment and formal ontology

To recognize an entailment relation between sentences requires one to grasp a piece of world knowledge that is not explicitly delivered in given premises. To formally capture the relevant knowledge, we focus on two types of semantic links between concepts: *is-a* links and *part-of* links.

The so-called monotonicity inference (Icard and Moss 2014) is a typical instance of inferences that require knowledge expressed by *is-a* links. For example, to derive an inference *A Eurostar runs* ⇒ *A train runs*, we need to rely on the knowledge that a Eurostar is a train.

*Part-of* links are used to describe parthood relations between concepts. An important class of inferences that depend on *part-of* links is *bridging inference* (Clark 1975):

(1) John got on a <u>Eurostar</u> and wanted to eat dinner. But <u>the buffet car</u> was not open.

To establish an anaphoric relation between the underlined noun phrases, one needs to use the knowledge that a buffet car is *part of* a Eurostar. What plays a crucial role in deriving such a bridging inference is *role concept* (Mizoguchi 2004). In contrast to basic concepts such as *train* and *human*, concepts like *buffet car*, *brake* and *passenger* represent a role in the context determined by the concept *train*. In general, a bridging inference is triggered by the expression denoting a role concept in a semantic representation; then the antecedent of the anaphora is identified with the concept that provides a context for the role concept. We will provide a formal representation of bridging inferences below.

To derive entailment relations and resolve anaphoric dependencies, one needs to give a formal description of world knowledge and then combine it with semantic representations (SRs) for the premises and conclusion. In this paper, we use a framework of DTS for building semantic representations. The entire system of building semantic representations and deriving entailment relations is pictured in Fig. 1.

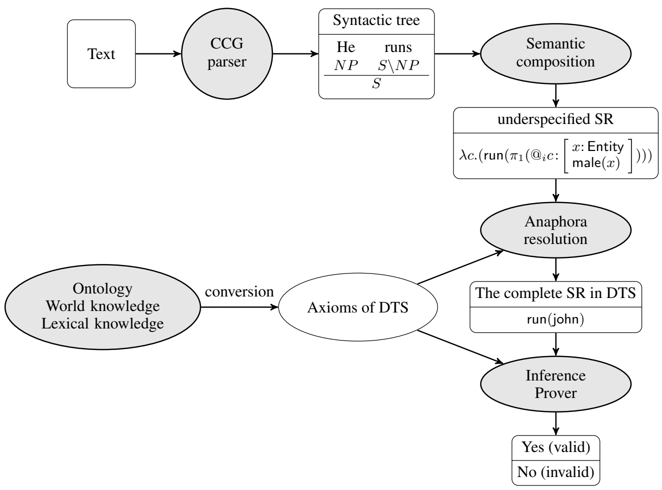



Figure 1: The entire inference system based on DTS

We use Combinatory Categorial Grammar (CCG) as a syntactic framework (Steedman 2000; Bekki 2010). Sentences that serve as inputs to the system are mapped to semantic representations in DTS via syntactic processing and semantic composition in CCG. A relevant piece of world knowledge and lexical knowledge is converted into axioms of DTS, and together with the outputs of semantic composition, they are used to derive the inferences in question. We will briefly introduce the framework of CCG and DTS.

## Combinatory Categorial Grammar (CCG)

CCG is based on the idea of *surface compositionality*, i.e., the idea of providing a compositional derivation of semantic representations based on surface structures of sentences.

There are two kind of syntactic categories, *basic categories* like $S$, $N$ and $NP$ and *complex categories* of the forms $X/Y$ and $X\backslash Y$. Complex categories of the form $X/Y$ expect their argument $Y$ to their right, while those of the form $X\backslash Y$ expect $Y$ to their left. Categories are combined by an application of combinatory rules. Each rule serves as a meaning composition rule, specifying how to combine the meanings (semantic representations) of expressions into a larger unit. In Fig. 2, the combinatory rule ($>$) on the left means that an expression having a category $X/Y$ and a meaning $f$, combined with an expression having a category $Y$ and a meaning $a$, yields an expression having a category $X$ and a meaning $fa$. Each meaning is represented as a $\lambda$-term. By virtue of this clear connection between syntax and semantics, CCG is particularly suitable for implementing compositional semantics.

$$\frac{X/Y : f \quad Y : a}{X : fa} > \qquad \frac{Y : a \quad X\backslash Y : f}{X : fa} <$$

Figure 2: Combinatory rules in CCG. Forward ($>$) and backward ($<$) function application rules.

## Dependent Type Semantics (DTS)

DTS (Bekki 2014) is a computational framework of natural language semantics based on dependent type theory. DTS gives a semantic representation for natural language sentences using $\Pi$-types and $\Sigma$-types in dependent type theory. $\Pi$-types correspond to universal quantifiers, while $\Sigma$-types correspond to existential quantifiers. An object of type $(\Pi x : A)\, B(x)$ is a function $f$ such that for any object $a$ of type $A$, $fa$ has type $B(a)$. Implication $A \to B$ is a degenerate form of $\Pi$-type: when $x$ does not occur free in $B$, $(\Pi x : A)\, B(x)$ is written as $A \to B$.

An object of type $(\Sigma x : A)\, B(x)$ is a pair $(a, b)$ of an object $a$ of type $A$ and an object $b$ of type $B(a)$. Conjunction $A \wedge B$ is a degenerate form of $\Sigma$-type: when $x$ does not occur free in $B$, $(\Sigma x : A)\, B(x)$ is written as $A \wedge B$. Projection function $\pi_1$ and $\pi_2$ are defined for objects of $\Sigma$-types, with the computation rules: $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. See, e.g., Martin-Löf (1984) and Ranta (1994) for more details on $\Pi$-types and $\Sigma$-types.

In DTS, $\Pi$-types and $\Sigma$-types are written as follows:

| | **$\Pi$-type** | **$\Sigma$-type** |
|---|---|---|
| Standard notation | $(\Pi x : A)\, B(x)$ | $(\Sigma x : A)\, B(x)$ |
| DTS notation | $(x : A) \to B(x)$ | $\begin{bmatrix} x{:}A \\ B(x) \end{bmatrix}$ |
| if $x \notin fv(B)$ | $A \to B$ | $\begin{bmatrix} A \\ B \end{bmatrix}$ |

Here $fv(B)$ means the set of free variables occurring in $B$. DTS uses the following abbreviation for $\Sigma$-types.

$$\begin{bmatrix} x : A \\ B \\ C \end{bmatrix} \equiv_{\text{def}} \begin{bmatrix} x{:}A \\ \begin{bmatrix} B \\ C \end{bmatrix} \end{bmatrix}$$

Fig.3 shows introduction and elimination rules for $\Pi$-types and $\Sigma$-types.

Dependent type theory has been used as an alternative to dynamic semantics (Ranta 1994). To handle anaphoric relations in a proper way, we need a system that can formally represent a dynamic binding relation as exemplified in (2a). It is known that the standard first-order logic (FOL) is not suitable for this purpose (Groenendijk and Stokhof 1991). That is, if we analyze a pronoun as a variable and conjoin the two sentences by means of conjunction, then the second variable $x$ remains unbound as indicated in (2b).

(2) a. Someone entered. He smiled.
b. $(\exists x\, \mathsf{Entered}\,(x)) \wedge \mathsf{Smiled}\,(x)$
c. $(\Sigma u : (\Sigma x : \mathsf{Entity})\, \mathsf{Entered}\,(x))\, \mathsf{Smiled}\,(\pi_1 u)$

By contrast, using $\Sigma$-types we can provide a semantic representation as shown in (2c). Here, the first sentence corresponds to the expression $(\Sigma x : \mathsf{Entity})\, \mathsf{Entered}(x)$; by

$$\frac{(x:A)\to B : \mathit{type} \quad \begin{array}{c}\overline{x:A}\\ \vdots \\ M:B\end{array}}{\lambda x.M : (x:A)\to B}\ (\Pi I) \qquad \frac{M:(x:A)\to B \quad N:A}{MN : B[N/x]}\ (\Pi E) \qquad \frac{M:\begin{bmatrix} x:A \\ B\end{bmatrix}}{\pi_1(M):A}\ (\Sigma E) \qquad \frac{M:\begin{bmatrix} x:A \\ B\end{bmatrix}}{\pi_2(M):B[\pi_1(M)/x]}\ (\Sigma E)$$

Figure 3: Introduction ($I$) and Elimination ($E$) rule of $\Pi$-type and $\Sigma$-type in DTS

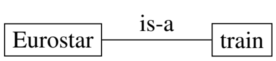
picking up a proof term of this type (= proposition) in terms of $\pi_1 u$ we can properly capture the anaphoric relation between the antecedent quantifier and the pronoun. We can also capture the accessibility constraints on anaphora (Karttunen 1969) in terms of inference rules for $\Sigma$-types and $\Pi$-types without any further stipulation. For details, see Bekki (2014).

A characteristic of DTS is that it is enriched with *underspecified terms* to give a fully compositional analysis of inferences involving anaphora. In the next section, we will apply this framework to the analysis of bridging inferences.

### Converting world knowledge into DTS formula

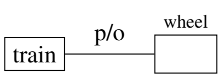
We will explain by examples how to convert world knowledge into DTS formulas. Concepts such as *train* and *wheel* in ontology correspond to 1-place predicates. The part-whole relation, "$x$ is part of $y$", corresponds to the 2-place predicate $x \preccurlyeq y$, where $\preccurlyeq$ is a partial order. *Is-a* links and *part-of* links are converted into DTS axioms in the following way. Here $\mathsf{Entity}$ represents the type of entities, and $\to$ is right-associative. We use the graphical notations for the semantic links (Mizoguchi and Kozaki 2009).

**1. is-a link**

A Eurostar is a train. 


**DTS axiom** $(x : \mathsf{Entity}) \to \mathsf{Eurostar}(x) \to \mathsf{Train}(x)$

**2. part-of link**

A wheel is part of a train. 


**DTS axiom** $(x : \mathsf{Entity}) \to \mathsf{Train}(x) \to \begin{bmatrix} y\colon \mathsf{Entity} \\ \mathsf{Wheel}(y) \\ y \preccurlyeq x \end{bmatrix}$

A *part-of* link has an axiom that derives the existence of the parts from the whole. It is known that FOL can be embedded into dependent type theory (Martin-Löf 1984), hence an entire family of description logic axioms (Baader 2003) that can be translated into FOL can also be converted to axioms in dependent type theory.

## Bridging inferences

Bridging inferences are special in that the antecedent is *inferred* given the information explicitly provided in a previous discourse and some relevant world knowledge (Krahmer and Piwek 1999). Due to this inferential character, it is not straightforward to handle bridging inferences in standard dynamic theories of anaphora such as Discourse Representation Theory (van der Sandt 1992; Kamp and Reyle 1993; Geurts 1999).

In DTS, the process of searching the antecedent of an anaphoric pronoun, including the case of bridging inferences, is formulated as a process of proof search, so that it can treat anaphora resolution and inferences with world knowledge in a unified way. DTS is augmented with an operator @ to represent anaphoric elements.

As an illustration, consider the discourse in (1) simplified as follows:

(3) a. John got on a Eurostar.
    b. But the buffet car was not open.

(3a) and (3b) are translated as shown in Fig. 4. Lexical entries and CCG-derivations for these sentences are provided in Appendix. In Fig. 4, the two semantic representations (SRs) are conjoined using dynamic conjunction $M\,;N$, which is defined as follows:

$$M\,;N \equiv_{\text{def}} \lambda c. \begin{bmatrix} u\colon Mc \\ N(c,u) \end{bmatrix}$$

Here $c$ is a proof term that encodes the information provided by the previous discourse; then, the pair $(c,u)$, where $u$ is the proof term for the SR of the first sentence, is passed to the SR of the second sentence. This context-passing mechanism allows us to handle dynamic binding. The bridging inference is triggered by the expression standing for the role concept *buffet car*, which is represented in terms of an @-operator. In the conjoined SR in Fig. 4, we have the following formula:

(4) $$\neg\mathsf{Open}(\pi_1(@_1(c,v) : \begin{bmatrix} y\colon \mathsf{Entity} \\ \mathsf{BuffetCar}(y) \\ y \preccurlyeq @_2(c,v) : \mathsf{Entity} \end{bmatrix}))$$

A term of the form $@_i c : \Lambda$ is called *type annotation*, where $\Lambda$ specifies the type of the term $@_i c$. The underspecified term $@_i$ is a function that takes a local context $c$ as argument. In the case of (4), the term $@_2(c,v)$ is annotated with the type $\mathsf{Entity}$ and the term $@_1(c,v)$ with the type corresponding to the proposition "there is a buffet car which is part of $@_2(c,v)$" represented as a $\Sigma$-type.

To begin with, we resolve the embedded @-term, that is, $@_2$. The underspecified term $@_2$ takes a local context $(c,v)$ as argument and returns an entity. The type of the @-operator can be specified in terms of type inferences; for instance, the type of $@_2$ can be specified in the following way:

(5) $$@_2 : \begin{bmatrix} \gamma \\ \begin{bmatrix} u\colon \begin{bmatrix} x\colon \mathsf{Entity} \\ \mathsf{Eurostar}(x) \end{bmatrix} \\ \mathsf{GetOn}(j, \pi_1 u) \end{bmatrix} \end{bmatrix} \to \mathsf{Entity}$$

Here $\gamma$ is the type of the previous context $c$. The process of resolving anaphora is a process to construct a proof term

$$\underbrace{\lambda c.\begin{bmatrix} u\colon \begin{bmatrix} x\colon \mathsf{Entity} \\ \mathsf{Eurostar}(x) \end{bmatrix} \\ \mathsf{GetOn}(j, \pi_1 u) \end{bmatrix}}_{\textbf{The SR of the sentence (3a)}} ; \underbrace{\lambda c.\neg\mathsf{Open}(\pi_1(@_1 c : \begin{bmatrix} y\colon \mathsf{Entity} \\ \mathsf{BuffetCar}(y) \\ y \preccurlyeq @_2 c : \mathsf{Entity} \end{bmatrix}}_{\textbf{The SR of the sentence (3b)}})) = \lambda c.\begin{bmatrix} v\colon \begin{bmatrix} u\colon \begin{bmatrix} x\colon \mathsf{Entity} \\ \mathsf{Eurostar}(x) \end{bmatrix} \\ \mathsf{GetOn}(j, \pi_1 u) \end{bmatrix} \\ \neg\mathsf{Open}(\pi_1(@_1(c,v) : \begin{bmatrix} y\colon \mathsf{Entity} \\ \mathsf{BuffetCar}(y) \\ y \preccurlyeq @_2(c,v) : \mathsf{Entity} \end{bmatrix})) \end{bmatrix}$$

Figure 4: The semantic representations of (3) in DTS

having this type and then replace the @-term by it. In this case, $\lambda c'.\pi_1(\pi_1(\pi_2 c'))$ gives a suitable term having the type in (5).

The $@_1$ is a term which requires, given the context $(c, v)$, an object $x$ of type $\mathsf{Entity}$ and a proof that $x$ is a buffet car that is part of the entity given by the $@_2(c, v)$. The intended anaphoric relation can be obtained by picking out the first element by means of the projection $\pi_1$. The type of the $@_1$ can be specified in a similar way to (5).

The knowledge required for the present example is given by the part-of relation "Eurostar has a buffet car", which is translated into the DTS formula (with a proof term $w$):

$$w : (x : \mathsf{Entity}) \to \mathsf{Eurostar}(x) \to \begin{bmatrix} y\colon \mathsf{Entity} \\ \mathsf{BuffetCar}(y) \\ y \preccurlyeq x \end{bmatrix}$$

Using this knowledge, we can construct a proof term in question and substitute it for the $@_1$-term. Then we can obtain the following semantic representation for the whole discourse in (3).

(6) $$\lambda c.\begin{bmatrix} v\colon \begin{bmatrix} u\colon \begin{bmatrix} x\colon \mathsf{Entity} \\ \mathsf{Eurostar}(x) \end{bmatrix} \\ \mathsf{GetOn}(j, \pi_1 u) \end{bmatrix} \\ \neg\mathsf{Open}(\pi_1(w(\pi_1(\pi_1 v))(\pi_2(\pi_1 v)))) \end{bmatrix}$$

## Inferences with total and partial predicates

In this section, we focus on adjectival predicates as an instance of predicate that requires the interaction between lexical knowledge and world knowledge. A pair of antonyms like *dirty* and *clean* is known to enable inferences depending on part-whole relations (Yoon 1996).

(7) a. Are the toys dirty?
 b. Are the toys clean?

If some of the toys are dirty, the answer to the question in (7a) should be "yes". In the case of (7b), by contrast, unless all of the toys are clean, the answer should be "no". We call predicates like *dirty* "partial predicates", and predicates like *clean* "total predicate". Fig. 5 shows typical examples of total predicates and the corresponding partial predicates. For example, as for the pair of *open* and *closed*, we can judge that a door is open when it is slightly open, while we usually judge that a door is closed only when it is completely closed.

There is a test which can be applied to distinguish between total and partial predicates (Rotstein and Winter 2004; Kennedy and McNally 2005). In normal contexts, it is odd to combine a partial predicate with a modifier such as *almost*, while it is perfectly fine with a total predicate.

| **partial predicate** | **total predicate** |
|---|---|
| dangerous | safe |
| dirty | clean |
| open | closed |
| wet | dry |
| sick | healthy |
| incomplete | complete |
| fail | pass |

Figure 5: Examples of total and partial predicates

(8) a. It is almost clean/safe. **total predicate**
 b. ?It is almost dirty/dangerous. **partial predicate**

By contrast, a modifier like *slightly* can naturally co-occur with partial predicates, while it requires a special context to co-occur with total predicates.

(9) a. ?It is slightly clean/safe. **total predicate**
 b. It is slightly dirty/dangerous. **partial predicate**

Another characteristic properly of total and partial predicates is concerned with entailment patterns: for a pair of antonyms of total and partial predicates, not only the assertion ($P$) of one form entails the negation ($\neg N$) of the other, but also the negation ($\neg N$) of one form entails the assertion ($P$) of the other (Kennedy and McNally 2005).

(10) a. clean ⇒ not dirty $P \Rightarrow \neg N$
 b. dirty ⇒ not clean $P \Rightarrow \neg N$

(11) a. not clean ⇒ dirty $\neg P \Rightarrow N$
 b. not dirty ⇒ clean $\neg P \Rightarrow N$

In the case of standard degree adjectives such as *small*, the assertion ($P$) of one form entails the negation ($\neg N$) of the other, but the converse does not hold; thus, *not small* does not entail *large* and *not large* does not entail *small*. For the proper treatment of inferences with lexical knowledge, it is important to specify such differences in entailment patterns for antonyms.

The lexical knowledge concerning partial and total predicates is conversed into DTS axioms as follows. Here $\mathsf{E}$ is an abbreviation for $\mathsf{Entity}$.

**total predicate**
$(x : \mathsf{E}) \to (y : \mathsf{E}) \to \mathsf{clean}(x) \to y \preccurlyeq x \to \mathsf{clean}(y)$

**partial predicate**
$(x : \mathsf{E}) \to (y : \mathsf{E}) \to \mathsf{dirty}(x) \to x \preccurlyeq y \to \mathsf{dirty}(y)$

A total predicate holds of a part if it holds of its whole; by contrast, a partial predicate holds of the whole if it holds

$$
\dfrac{
  \dfrac{
    \dfrac{\dfrac{}{q : \begin{bmatrix} \mathsf{window}(x) \\ x \preccurlyeq a \end{bmatrix}}\;(1)}{\pi_2 q : x \preccurlyeq a}\;(\Sigma E)
    \qquad
    \dfrac{
      \dfrac{\overset{\textsf{Premise}}{p : \begin{bmatrix} \mathsf{train}(a) \\ \mathsf{clean}(a) \end{bmatrix}}}{\pi_2 p : \mathsf{clean}(a)}\;(\Sigma E)
      \qquad
      \dfrac{
        \dfrac{}{x : \mathsf{E}}\;(2)
        \qquad
        \dfrac{\overset{\textsf{Axiom for the total predicate “clean”}}{a : \mathsf{E} \quad f : (x : \mathsf{E}) \to (y : \mathsf{E}) \to \mathsf{clean}(a) \to x \preccurlyeq a \to \mathsf{clean}(x)}}{fa : (y : \mathsf{E}) \to \mathsf{clean}(a) \to y \preccurlyeq a \to \mathsf{clean}(y)}\;(\Pi E)
      }{fax : \mathsf{clean}(a) \to x \preccurlyeq a \to \mathsf{clean}(x)}\;(\Pi E)
    }{fax(\pi_2 p) : x \preccurlyeq a \to \mathsf{clean}(x)}\;(\Pi E)
  }{
    \dfrac{fax(\pi_2 p)(\pi_2 q) : \mathsf{clean}(x)}{\lambda x.fax(\pi_2 p)(\pi_2 q) : \begin{bmatrix} \mathsf{window}(x) \\ x \preccurlyeq a \end{bmatrix} \to \mathsf{clean}(x)}\;(\Pi I),(1)
  }\;(\Pi E)
}{\lambda q \lambda x.fax(\pi_2 p)(\pi_2 q) : (x : \mathsf{E}) \to \begin{bmatrix} \mathsf{window}(x) \\ x \preccurlyeq a \end{bmatrix} \to \mathsf{clean}(x)}\;(\Pi I),(2)
$$

Figure 6: A derivation of the inference in (12). We omit derivation of $type$ condition with $\Pi$ type.

of its part. The entailment patterns of antonym pairs can be converted in a similar way; we omit the details here.

As an illustration of how an inference is derived using lexical and world knowledge, consider the following example:

(12)
| | |
|---|---|
| Premise : | The train is clean. |
| Conclusion : | The windows are clean. |

Fig.6 shows a derivation of this inference. Here, the subject *train* in the premise is represented by the constant $a$ of type $\mathsf{E}$, thus we assume $a : \mathsf{E}$ in the derivation. The subject *the windows* in the conclusion has to be interpreted as *the train's windows* via a bridging inference. For simplicity, we assume a semantic representation after resolving the bridging for the conclusion in (12). The axiom for total predicates is available for the predicate *clean*: we use the term $f$ for the proof term of this axiom.

Predicates *functioning* and *broken* are antonyms that can be taken as total and partial predicates, respectively; given that a brake is part of a train, the following inferences are naturally taken to be valid, while the opposite directions are not.

*The train is functioning* $\Rightarrow$ *The brake is functioning*
*The brake is broken* $\Rightarrow$ *The train is broken*

However, the same inference does not hold for the following pair of examples, even if a window is part of a train.

*The train is functioning* $\not\Rightarrow$ *The window is functioning*
*The window is broken* $\not\Rightarrow$ *The train is broken*

This suggests that the proper treatment of inferences with predicates *functioning* and *broken* will involve some additional complexity. A detailed investigation of this issue is left for another occasion.

## Conclusion

In this paper, we have presented a framework combining DTS with formal ontology to handle inferences involving knowledge of part-whole relations. We have showed how to formalize bridging inferences and inferences with partial and total predicates, both being sensitive to part-whole relations among structured objects.

The inference component in our framework can be implemented using a modern proof assistant based on dependent type theory such as Coq, which has been applied to the formalization of natural language inferences (Chatzikyriakidis and Luo 2014). All the forms of proofs discussed in this paper can be verified and automated using a relatively simple combination of tactics in Coq. Further research on proof automation as well as exploring other forms of lexical and world knowledge such as described by *attribute-of* relations is left for future work.

| Expression | *Syntactic category* : *Semantic representation* |
|---|---|
| John | $NP : j$ |
| got on | $VP\backslash NP : (\lambda yxc)\mathsf{GetOn}(x, y)$ |
| a | $VP\backslash(VP/NP)/N : (\lambda npxc)\begin{bmatrix} u : \begin{bmatrix} y : \mathsf{Entity} \\ nyc \end{bmatrix} \\ p(\pi_1 u)xc \end{bmatrix}$ |
| Eurostar | $N : (\lambda xc)\mathsf{Eurostar}(x)$ |
| the | $(S/VP)/N : (\lambda npc)(p\pi_1(@_i c) : \begin{bmatrix} x : \mathsf{Entity} \\ nxc \end{bmatrix})$ |
| buffet car | $N : (\lambda xc)\begin{bmatrix} \mathsf{BuffetCar}(x) \\ x \preccurlyeq @_i c : \mathsf{Entity} \end{bmatrix}$ |
| was | $VP/VP : (\lambda pxc)(pxc)$ |
| not | $VP/VP : (\lambda pxc)(\neg pxc)$ |
| open | $VP : (\lambda xc)\,\mathsf{Open}(x)$ |

Figure 7: Lexical entries for the example (3) in DTS


## Acknowledgements

We thank the three anonymous reviewers for helpful comments and suggestions. We also thank Riichiro Mizoguchi and Masaharu Yoshioka for useful discussions. This work was supported by MEXT/JSPS KAKENHI Grant Number #15K00301 and #26770009.


## Appendix

Fig. 7 provides lexical entires for the example in (3). Here $VP$ is an abbreviation of the syntactic category $S\backslash NP$ in CCG. Compositional derivations of (3a) and (3b) are given in Fig. 8 and Fig. 9, respectively.

## References


Baader, F. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge university press.

$$\dfrac{\text{John}}{NP : j} \qquad \dfrac{\text{got on}}{(S\backslash NP)/NP : (\lambda yxc)\mathsf{GetOn}(x,y)} \qquad \dfrac{\text{a}}{(S\backslash NP)\backslash((S\backslash NP)/NP)/N : (\lambda npxc)\begin{bmatrix} u : \begin{bmatrix} y : \mathsf{Entity} \\ nyc \end{bmatrix} \\ p(\pi_1 u)xc \end{bmatrix}} \qquad \dfrac{\text{Eurostar}}{N : (\lambda xc)\mathsf{Eurostar}(x)}$$

$$\dfrac{(S\backslash NP)\backslash((S\backslash NP)/NP)/N \quad N}{(S\backslash NP)\backslash((S\backslash NP)/NP) : (\lambda pxc)\begin{bmatrix} u : \begin{bmatrix} y : \mathsf{Entity} \\ \mathsf{Eurostar}(y) \end{bmatrix} \\ p(\pi_1 u)xc \end{bmatrix}} >$$

$$\dfrac{(S\backslash NP)/NP \quad (S\backslash NP)\backslash((S\backslash NP)/NP)}{S\backslash NP : (\lambda xc)\begin{bmatrix} u : \begin{bmatrix} y : \mathsf{Entity} \\ \mathsf{Eurostar}(y) \end{bmatrix} \\ \mathsf{GetOn}(x, \pi_1 u) \end{bmatrix}} <$$

$$\dfrac{NP : j \quad S\backslash NP}{S : \lambda c.\begin{bmatrix} u : \begin{bmatrix} y : \mathsf{Entity} \\ \mathsf{Eurostar}(y) \end{bmatrix} \\ \mathsf{GetOn}(j, \pi_1 u) \end{bmatrix}} <$$

Figure 8: A compositional derivation of the SR of (3a)

$$\dfrac{\text{The}}{(S/(S\backslash NP))/N : (\lambda npc)(p\pi_1(@_1 c)c : \begin{bmatrix} x : \mathsf{Entity} \\ nxc \end{bmatrix})} \qquad \dfrac{\text{buffet car}}{N : (\lambda xc)\begin{bmatrix} \mathsf{BuffetCar}(x) \\ x \preccurlyeq @_2 c : \mathsf{Entity} \end{bmatrix}}$$

$$\dfrac{(S/(S\backslash NP))/N \quad N}{S/(S\backslash NP) : (\lambda pc)(p\pi_1(@_1 c)c : \begin{bmatrix} x : \mathsf{Entity} \\ \mathsf{BuffetCar}(x) \\ x \preccurlyeq @_2 c : \mathsf{Entity} \end{bmatrix})} >$$

$$\dfrac{\text{was}}{(S\backslash NP)/(S\backslash NP) : (\lambda pxc)(pxc)} \qquad \dfrac{\text{not}}{(S\backslash NP)/(S\backslash NP) : (\lambda pxc)\,\neg pxc} \qquad \dfrac{\text{open}}{S\backslash NP : (\lambda xc)\,\mathsf{Open}(x)}$$

$$\dfrac{(S\backslash NP)/(S\backslash NP) \quad S\backslash NP}{S\backslash NP : (\lambda xc)\,\neg\mathsf{Open}(x)} >$$

$$\dfrac{(S\backslash NP)/(S\backslash NP) \quad S\backslash NP}{S\backslash NP : (\lambda xc)\,\neg\mathsf{Open}(x)} >$$

$$\dfrac{S/(S\backslash NP) \quad S\backslash NP}{S : \lambda c.\neg\mathsf{Open}(\pi_1(@_1 c) : \begin{bmatrix} x : \mathsf{Entity} \\ \mathsf{BuffetCar}(x) \\ x \preccurlyeq @_2 c : \mathsf{Entity} \end{bmatrix})} >$$

Figure 9: A compositional derivation of the SR of (3b)


Bekki, D. 2010. *A Formal Theory of Japanese Grammar: The Conjugation System, Syntactic Structures, and Semantic Composition*. Kuroshio.

Bekki, D. 2014. Representing anaphora with dependent types. In *LACL 2014*, volume 8535 of *Lecture Notes in Computer Science*. Springer. 14–29.

Bos, J. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, 277–286.

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

Clark, S., and Curran, J. R. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4):493–552.

Clark, H. H. 1975. Bridging. In *Theoretical issues in natural language processing*, 169–174.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Gangemi, A.; Guarino, N.; Masolo, C.; and Oltramari, A. 2003. Sweetening WordNet with DOLCE. *AI magazine* 24(3):13.

Geurts, B. 1999. *Presuppositions and Pronouns*. Elsevier.

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

Icard, T., and Moss, L. 2014. Recent progress in monotonicity. *Linguistic Issues in Language Technology* 9.

Kamp, H., and Reyle, U. 1993. *From Discourse to Logic*. Reidel.

Karttunen, L. 1969. Discourse referents. In *Proceedings of the 1969 conference on Computational linguistics*, 1–38.

Kennedy, C., and McNally, L. 2005. Scale structure, degree modification, and the semantics of gradable predicates. *Language* 81(2):345–381.

Krahmer, E., and Piwek, P. 1999. Presupposition projection as proof construction. In Bunt, H., and Muskens, R., eds., *Computing Meaning*, volume 1. Kluwer. 281–300.

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

Mizoguchi, R., and Kozaki, K. 2009. Ontology engineering environments. In Staab, S., and Studer, R., eds., *Handbook on Ontologies*. Springer Berlin Heidelberg. 315–336.

Mizoguchi, R. 2004. Part 3: Advanced course of ontological engineering. *New Generation Computing* 22(2):193–220.

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

Rotstein, C., and Winter, Y. 2004. Total adjectives vs. partial adjectives: Scale structure and higher-order modifiers. *Natural Language Semantics* 12(3):259–288.

Steedman, M. 2000. *The Syntactic Process*. MIT Press.

van der Sandt, R. A. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics* 9:333–377.

Yoon, Y. 1996. Total and partial predicates and the weak and strong interpretations. *Natural Language Semantics* 4(3):217–236.