Encoding Generalized Quantifiers in Dependency-based Compositional Semantics

Yubing Dong[*]
Department of Computer Science
University of Southern California
yubing.dong@usc.edu

Ran Tian
Graduate School of Information Sciences
Tohoku University
tianran@ecei.tohoku.ac.jp

Yusuke Miyao
National Institute of Informatics, Japan
yusuke@nii.ac.jp

Abstract

For textual entailment recognition systems, it is often important to correctly handle Generalized quantifiers (GQ). In this paper, we explore ways of encoding GQs in a recent framework of Dependency-based Compositional Semantics, especially aiming to correctly handle linguistic knowledge like hyponymy when GQs are involved. We use both the *selection operator* mechanism and a new *relation* extension to implement some major properties of GQs, reducing 69% errors of a previous system, and a further error analysis suggests extensions towards more powerful logical systems.

1 Introduction

Dependency-based Compositional Semantics (DCS) provides a formal yet intuitive way to model natural language semantics. It was initially proposed in Liang et al. (2011) as a relational database querying protocol, and later used for logical inference in Tian et al. (2014a). Although the DCS inference framework provided decent support for both quantifiers *all* (universal quantifier) and *no* (negated existential quantifier), attention is required for an RTE system to cope with generalized quantifiers (GQ), including "at most n", "at least n", "most", etc., which can affect the direction or even the existence of an entailment relation, as demonstrated in Examples 1 to 3.

[*] This work was conducted during an internship at the National Institute of Informatics, Japan.

Example 1. $P \Rightarrow H$ but $H \nRightarrow P$, where
P At most 5 students like noodles.
H At most 5 Japanese students like udon noodles.

Example 2. $P \Rightarrow H$ but $H \nRightarrow P$, where
P At least 5 Japanese students like udon noodles.
H At least 5 students like noodles.

Example 3. $P \nRightarrow H$ and $H \nRightarrow P$, where
P Most Japanese students like udon noodles.
H Most students like noodles.

In this paper, we explore ways of encoding GQs in a recent framework of Dependency-based Compositional Semantics (DCS) (Liang et al., 2013; Tian et al., 2014a), especially aiming to correctly handle linguistic knowledge like hyponymy when GQs are involved. We use *selection operators*, an extension mechanism described in Tian et al. (2014a), to implement a sub-type of GQs (Section 3.1). To deal with downward monotonicity of the predicate argument, we also propose a simple extension called "*relation*" to the framework (Section 3.2). This approach does not encode the exact semantics of every specific GQ, but instead captures some major properties that are both easily implementable with the current technology and useful in many cases.

As in Tian et al. (2014a), we empirically tested the extended system on the "Generalized Quantifiers" section of the FraCaS corpus (The Fracas Consortium et al., 1996), and reduced 69% of the previous errors. A further error analysis reveals some limitations of the current approach, suggesting extensions towards more powerful logical systems. We hope this research could make linguistic knowledge like hyponymy a more effective resource for textual entailment tasks, and also shed some light on the handling of more complicated natural language inference phenomena. The extended system is publicly released at `https://github.com/tomtung/tifmo`.

2 Background

2.1 Properties of Generalized Quantifiers

In this paper, "*generalized quantifiers*" refers to quantity-denoting determiners such as *"few"*, *"most"*, *"at least 5"*, etc. They can bind with a property-denoting common noun phrase (e.g. *"students"*) to form a quantified noun phrase (e.g. *"few students"*), which can then bind with a predicate (e.g. *"like noodles"*) to form a sentence (e.g. *"few students like noodles"*). We regard the meanings of both the common noun phrase and the predicate as their *denotations*, i.e. let W be the universe containing all entities (a.k.a. the "world" set), then the meaning of "*students*" is regarded as a set **student** $\subseteq W$ containing all entities being students, and the meaning of "*(someone) likes noodles*" is regarded as a subset of W which contains all entities who like noodles. Thus, if we denote the power set of W as 2^W, then a GQ can be seen as a binary relation over 2^W, or in other words, a function F from $(A, B) \in 2^W \times 2^W$ to $F(A)(B) \in \mathbf{2} = \{0, 1\}$, in which the sets A and B are called *noun argument* and *predicate argument*, respectively.

Usually, the relation imposed by a GQ is based on the notion $|\cdot|$ of set cardinalities; for example, $AtMost[30\%](A)(B)$ represents the relation that $|A \cap B|/|A| \leq 30\%$. However, in practice it often requires a large amount of effort to introduce cardinalities into logical inference. Hence, in this paper we make a compromise by encoding properties of GQs that are most relevant to semantic relations like hyponymy and are useful for solving RTE problems. We mainly focus on three major properties, namely *interaction with universal and existential quantifications*, *conservativity*, and *monotonicity*.

Given a GQ, say F, one most basic semantic property is its interaction with universal and existential quantifications—whether $F(A)(B)$ is entailed by the noun argument being a subset of the predicate argument (for short, "*entailed by* $\forall$"), i.e. $A \subseteq B \Rightarrow F(A)(B)$, or whether it entails the two arguments having a non-empty intersection (for short, "*entails* $\exists$"), i.e. $F(A)(B) \Rightarrow A \cap B \neq \emptyset$. There are three cases:

$$A \subseteq B \Rightarrow F(A)(B) \Rightarrow A \cap B \neq \emptyset$$

as *"most"* in Example 4,

$$A \subseteq B \not\Rightarrow F(A)(B) \Rightarrow A \cap B \neq \emptyset$$

as *"a lot of"* in Example 5, and

$$A \subseteq B \not\Rightarrow F(A)(B) \not\Rightarrow A \cap B \neq \emptyset$$

as *"at most 5"* in Example 6.[1]

Example 4. $A \Rightarrow B \Rightarrow C$, where
A All students like noodles.
B Most students like noodles.
C There are students who like noodles.

Example 5. $A \not\Rightarrow B \Rightarrow C$, where
A All students like noodles.
B A lot of students like noodles.
C There are students who like noodles.

Example 6. $A \not\Rightarrow B \not\Rightarrow C$, where
A All students like noodles.
B At most 5 students like noodles.
C There are students who like noodles.

The *conservativity* property of GQs results from the "domain restraining" role of the noun argument, which effectively eliminates objects that do not have the noun property, so that we only need to consider which of the rest has the predicate property. For example:

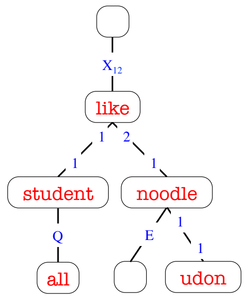
Example 7. Few apples are toxic. $\Longleftrightarrow$ Few apples are toxic apples.

The intuition here is that to know whether *few apples* are *toxic*, it is sufficient to know which *apples* are *toxic*; those non-apple toxicants are irrelevant. We formally define the conservativity property as follows.

Definition 1 (Conservativity). A GQ F is *conservative* if for any $A, B \subseteq W$,

$$F(A)(B) \Longleftrightarrow F(A)(A \cap B).$$

[1]We have made a convenient and practical assumption here: for an English GQ denoted as $F(\cdot)(\cdot)$, $F(A)(B)$ presupposes $A \neq \emptyset$. Therefore we ignore the cases when $A \subseteq B \Rightarrow F(A)(B) \not\Rightarrow A \cap B \neq \emptyset$, because $A \neq \emptyset \wedge A \subseteq B \Rightarrow A \cap B \neq \emptyset$.



student

Tom
Kate
John
...

like

Mary	Teddy bear
John	kitsune udon
Mike	spaghetti
...	...

udon

kitsune udon
tempura udon
...

noodle

kitsune udon
spaghetti
...

Figure 1: A DCS tree of the sentence *"all students like udon noodles"*, with a given database.

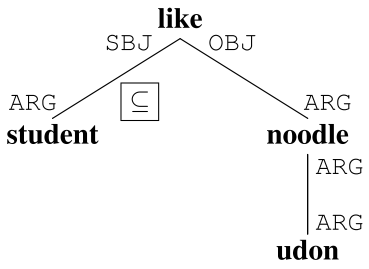


Figure 2: Adapted DCS tree for logical inference

Another important property that textual entailment could rely on is *monotonicity*.

Definition 2 (Monotonicity)**.** A GQ F is *upward-entailing* (resp. *downward-entailing*) in the noun argument if, for any $A, B \subseteq W$ and $A' \subseteq A$ (resp. $A' \supseteq A$),

$$F(A')(B) \Rightarrow F(A)(B).$$

F being *upward/downward-entailing* in the predicate argument can be defined in a similar manner.

For example, the GQ *"at most 5"* is downward-entailing in each argument as shown in Example 1; and the GQ *"at least 5"* is upward-entailing as shown in Example 2.

In Section 3, we will explore ways of encoding the properties discussed in this section in the DCS inference framework.

2.2 Dependency-based Compositional Semantics

DCS (Liang et al., 2013) was originally proposed as a natural language interface for querying concrete relational databases. The meanings of a natural language sentence in DCS are represented by a *DCS tree*, which is designed to be both semantically precise for execution on a database, and structurally straightforward for easy alignment to a syntactic dependency tree. For example, Figure 1 shows the DCS tree for the sentence "all students like udon noodles", with the corresponding tables in a given relational database.

When executed on databases, a DCS tree calculates *denotations* in a bottom-up manner. For example, the DCS tree in Figure 1 first takes the table `student`, and stores it at place "1"; then it calculates the intersection of entries in table `udon` and `noodle` to get the denotation of *"udon noodles"*, and stores the result at place "2". The execute marker "X_{12}" on the root edge imposes the wide reading of the quantifier *"all"*, and guides a calculation that first joins the result stored at place "2" with the second column of `like` table, producing the denotation of *"like udon noodles"*; then projects this denotation into the first column to get the denotation of *"subjects who like udon noodles"*; and finally checks if this is a superset of the denotation of *"students"* stored in place "1". The narrow reading of *"all"* (i.e. "there is a specific udon noodle liked by all students") can be produced by replacing the execute marker X_{12} with X_{21}, which will first assembles each entry x in the second column of the `like` table such that all students like x, then intersects the result with the denotation of *"udon noodles"*, and finally checks if the intersection is an empty set. For the precise calculation of a DCS tree and details of this "mark-execute" mechanism, please consult Liang et al. (2013).

In Tian et al. (2014a), the DCS framework was adapted to deal with open-domain textual inference. The idea is to use relational algebra operators (Codd, 1970) to formalize the calculation process used in DCS trees, so we can perform logical inference on this abstract level, without given a closed-domain relational database. For example, Figure 2 shows an adapted DCS tree representing the same sentence, *"all students like udon noodles"*; and it guides a calculation of the meaning parallel to the original DCS. Concretely, the *abstract denotation* of *"udon noodles"* is formulated as the following:

$$D_1 = \textbf{noodle} \cap \textbf{udon},$$

where **noodle** and **udon** are no longer given tables in a relational database but abstract sets (treated as symbols) representing denotations of the words, and "$\cap$" is a relational algebra operator representing "intersection".

Similarly, the abstract denotation of *"like udon noodles"* is formulated by:

$$D_2 = \textbf{like} \cap (W_{\texttt{SBJ}} \times (D_1)_{\texttt{OBJ}}),$$

where W is the "world set" as mentioned in Section 2.1, and "$\times$" denotes the Cartesian product. Subscripts SBJ and OBJ are used to denote different dimensions.

Finally, the abstract denotation of *"subjects who like udon noodles"* is:

$$D_3 = \pi_{\texttt{SBJ}}(D_2),$$

where π is the projection operator. Here we use π_r to denote a projection into dimension r, whereas π^r denotes a projection to all dimensions other than r.

The adapted DCS tree in Figure 2 uses syntactic/semantic labels (SBJ, OBJ, etc.) instead of numbers in the original DCS tree to denote different dimensions (i.e. different columns in the tables of the relational database), because they provide database-independent explanations for these dimensions. In addition, the involved "mark-execute" mechanism for representing quantifier *"all"* (as illustrated by the Q, E and X_{12} markers in Figure 1) is simplified to a quantification marker "$\subseteq$" on the **student**-**like** edge (Figure 2), and explained as the division operator $q_{\subseteq}$ in relational algebra[2]:

$$q^r_{\subseteq}(R, C) = \{x |\ \emptyset \neq R \cap (\{x\} \times W_r) \subseteq \{x\} \times C_r\}$$

Therefore, the abstract denotations

$$\begin{aligned} D_4 &= q^{\texttt{SBJ}}_{\subseteq}\left(\pi^{\texttt{OBJ}}(D_2), \textbf{student}\right) \\ &= q^{\texttt{SBJ}}_{\subseteq}(D_3, \textbf{student}) \end{aligned}$$

and

$$D_5 = \pi^{\texttt{OBJ}}\left(q^{\texttt{SBJ}}_{\subseteq}(D_2, \textbf{student})\right),$$

correspond to the final results calculated by the original DCS tree according to the wide reading and narrow reading of *"all"*, respectively. For logical inference, instead of the database-dependent evaluations of such denotations, we mainly consider their *satisfiability*, i.e. whether D_4 (or D_5) $\neq \emptyset$. Here, by definition of the division operator, $D_4 \neq \emptyset \Leftrightarrow \textbf{student} \subseteq D_3$ and $D_5 \neq \emptyset \Leftrightarrow q^{\texttt{SBJ}}_{\subseteq}(D_2, \textbf{student}) \neq \emptyset \Leftrightarrow \exists x; \textbf{student}_{\texttt{SBJ}} \times \{x\}_{\texttt{OBJ}} \subseteq D_2$.

As we can see from the previous description, many intermediate or related denotations are produced during the processing of DCS trees. In Tian et al. (2014a), a special kind of auxiliary denotations is considered, which integrates the context information of an entire DCS tree, and is naturally linked to a single pairing of a syntactic/semantic label and a node in the DCS tree. Such a pair is called a *germ*, denoted by $(\textbf{like}, \texttt{SBJ})_{\mathcal{T}}$, $(\textbf{like}, \texttt{OBJ})_{\mathcal{T}}$, $(\textbf{noodle}, \texttt{ARG})_{\mathcal{T}}$, etc., where the subscript $\mathcal{T}$ is used to denote the whole DCS tree and emphasize the context awareness of the germ object. Abstract denotations linked to germs are closely related to the concept of *feasible values* defined in Liang et al. (2013). For example, if we consider the DCS tree $\mathcal{T}$ in Figure 2 and assume the wide reading of *"all"*, then the denotations linked to $(\textbf{like}, \texttt{OBJ})_{\mathcal{T}}$ and $(\textbf{noodle}, \texttt{ARG})_{\mathcal{T}}$ both equal to $\pi_{\texttt{OBJ}}(D_2) = \pi_{\texttt{OBJ}}(\textbf{like}) \cap D_1$, *"udon noodles that are liked by somebody"*; the denotation linked to $(\textbf{like}, \texttt{SBJ})_{\mathcal{T}}$ is D_3, *"subjects who like udon noodles"*; and the denotation linked to $(\textbf{student}, \texttt{ARG})_{\mathcal{T}}$ is **student**. The final result D_4 can then be seen as been calculated from the abstract denotations linked to germs $(\textbf{like}, \texttt{SBJ})_{\mathcal{T}}$ and $(\textbf{student}, \texttt{ARG})_{\mathcal{T}}$. Abstract denotations linked to germs are useful for encoding GQs in the DCS framework, as we describe in Section 3.2.

Another useful mechanism for implementing GQs is the *selection operator* s_f introduced in Tian et al. (2014a), which are marked on a DCS tree node and wrap an abstract denotation D to form a new abstract denotation $s_f(D)$ during the calculation process. Selection operators were introduced as an extension mechanism to represent the generalized selection operation in relational algebra, which selects a subset of specific properties from a given set; the axioms characterizing such properties can be user-defined. For example, in Tian et al. (2014a), selection operators are used to implement superlatives such as *"highest"*, so that $s_{\text{highest}}(\textbf{mountain})$ denotes the set of the highest mountains. Effectively, a selection operator s_f is a user-defined map from any

[2]When R and C have the same dimension, $q^r_{\subseteq}(R, C)$ is either the 0-dimension point set $\{*\}$ (if $R \subseteq C$) or (otherwise) $\emptyset$.

abstract denotation D to a new denotation $s_f(D)$. In Section 3.1 we will use this mechanism to encode GQs.

2.3 Related Work

GQs have been a topic of interest in study of logic since ancient time: Aristotle's syllogism could be seen as concerning the meanings and properties of four basic quantifiers, namely *"all"*, *"no"*, *"some"*, and *"not all"*. Gottlob Frege (Zalta, 2014), one of the founders of modern logic, in 1870s introduced $\forall$ and $\exists$, and formulated the notion of a quantifier as a second order relation. The idea of *generalized* quantifiers was introduced by Mostowski (1957) and generalized in Lindström (1966), forming the standard definition we use nowadays. Later, Barwise and Cooper (1981), following Montague (1973), showed the importance of GQs in the formal analysis of linguistic phenomena. By and large, these works cover the logical and linguistic background involved in this paper.

Although it has been recognized that it is important to encode GQs for solving textual entailment problems, this remains a big challenge. MacCartney et al. (2006), for example, tried to capture the use of GQs in feature vectors, but the capabilities of which are greatly limited without an inference engine. Even for systems that are backed by inference engines like in this paper, the focus still needs to be put on practical NLP rather than logic, linguistics, or semantic theory, and model complexity may need to be purposely traded for computation efficiency. For example, Lewis and Steedman (2013) used first-order logic for semantic representation, which is theoretically very expressive, but still unable to define GQs without some extensions (Barwise and Cooper, 1981) that are nontrivial especially for practical inference.

Some works made the compromise similar to ours: only encode the important properties of GQs rather than their perfect semantics. A notable recent work that focused on monotonicity is MacCartney and Manning (2008), in which the notion of monotonicity was generalized to support recursive determination of entailments of a compound expression from its constituents. To a large extent, this approach handled the interaction between multiple GQs in a single sentence. However, inference was based on a chain of shallow syntactic edit operations linking premise to hypothesis, which not only failed to include various inference patterns, but also was unreliable when there are multiple sentences in the premise, or when the premise is relatively long. The DCS inference framework, on the other hand, gracefully handles such cases in a uniform way, thanks to the more sophisticated inference engine. An empirical comparison is shown in Section 4.

The logical inference engine described in Tian et al. (2014a) treats abstract denotations as terms and represent meanings by atomic sentences, which is shown to be very efficient compared to first order logic provers (Tian et al., 2014b). The idea behind this is actually very similar to description logics (DL) (Baader et al., 2003); indeed, the $\mathcal{DLR}$ generalization of DLs towards n-ary relations (Calvanese et al., 1998) was proposed to deal with inference problems on database schemata expressed in relational models, which shares the same setting with the logical system proposed in Tian et al. (2014a). The $\mathcal{DLR}$ system includes intersections, Cartesian products of 1-dimensional sets and projections into 1-dimensional sets, as well as constructors not presented in Tian et al. (2014a)'s system such as complement, union, and qualified number restrictions. $\mathcal{DLR}$ is also shown to be reducible to the traditional DL (with binary relations) $\mathcal{ALCQI}$, for which many complete DL inference engines are available[3]. In comparison, Tian et al. (2014a)'s inference engine is not complete and lacks a thorough exploration from the theoretical side, e.g. on the decidability and complexity of the logical system, but it has a working natural language interface inherited from the DCS framework, and supports specific constructors tailored for textual inference, e.g. the division operator, which seems not easily encoded in $\mathcal{DLR}$. Description logics have been applied to natural language processing since the early days, but were used mostly for *semantic interpretation* (Brachman, 1985; Sowa, 1991; Knight and Luk, 1994), in which knowledge on syntactic, semantic, and pragmatic elements of natural language are encoded in DL to drive the process of converting utterances into deep and context-dependent logical

[3]`http://www.cs.man.ac.uk/~sattler/reasoners.html`

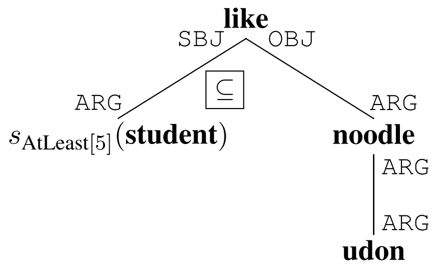


Figure 3: Encoding *"at least 5"* as selection

forms. Tian et al. (2014a)'s work on the other hand directly uses a DL-like logical system to represent semantics and perform textual inference, benefiting from the efficiency of DL logical inference. We explore ways of extending Tian et al. (2014a)'s system to deal with more advanced linguistic phenomena in this work, while trying to preserve its algebraic fashion to ensure efficiency, because we believe it is important to investigate to what extent the "natural" textual inference requires from a logical system. Some limitations of Tian et al. (2014a)'s framework has actually been revealed; for which we will discuss in Section 4.

3 Encoding Generalized Quantifiers

3.1 Encoding as Selections

Selections are used to encode GQs in the form of

$$F(A)(B) \equiv s_F(A) \subseteq B,$$

where s_F is the specific selection operator defined for the GQ F—that is, a selection operator s_F is always used together with a quantification marker "$\subseteq$", as exemplified in Figure 3. Note that $s_F(A)$ can be defined as any set related to A, not necessarily being its subset.

The basic requirement for encoding a GQ F in this way is that F should be upward-entailing in its predicate argument, because the form $s_F(A) \subseteq B$ implies such monotonicity. Entailment from universal quantification (Example 4)

$$A \subseteq B \Rightarrow s_F(A) \subseteq B$$

and conservativity

$$s_F(A) \subseteq (A \cap B) \Leftrightarrow s_F(A) \subseteq B$$

both hold if we add axiom:

$$s_F(A) \subseteq A$$

On the other hand, entailment to existential quantification (Example 5)

$$s_F(A) \subseteq B \Rightarrow A \cap B \neq \emptyset$$

can be implied from the custom axiom:

$$s_F(A) \cap A \neq \emptyset$$

The monotonicity in the noun argument can be implemented as well. If F is upward-entailing in the noun argument, we should add the axiom

$$A' \supseteq A \Rightarrow s_F(A') \subseteq s_F(A).$$

Note that the direction of $\subseteq$ is reversed because $s_F(A)$ serves as the *subset* in the form $F(A)(B) \equiv s_F(A) \subseteq B$. Similarly, downward-entailment in the noun argument can be achieved by the axiom

$$A' \subseteq A \Rightarrow s_F(A') \subseteq s_F(A).$$

A proof tree for Example 2 is shown in Figure 4, where D_3 is the denotation for *"subjects who like udon noodles"*, as defined in Section 2.2, and similarly $D'_3 = \pi_{\texttt{SBJ}}(\textbf{like} \cap (W_{\texttt{SBJ}} \times \textbf{noodle}_{\texttt{OBJ}}))$ for *"subjects who like noodles"*.

3.2 Encoding as Relations

As mentioned in Section 2.1, a GQ can be seen a binary relation over 2^W. From this point of view, we introduce a new extension called *relation* as a new type of statement into the framework. A relation r_f can be used to represent an arbitrary relation between two abstract denotations. A relation marker can be marked on a DCS tree edge to denote some relation between the child germ and the parent germ (Figure 5). In our implementation, the core inference engine keeps track of which term pairs are labeled with which relations: not only can it answer whether two terms have been claimed to have a certain relation, but also can it look up all terms that have a certain relation with a certain term. Similar to selections, we can also specify different sets of axioms for different relations—an axiom could be about either what a relation statement entails or what it is entailed by.

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Algebraic Property}}{\textbf{student} \supseteq \textbf{Japanese} \cap \textbf{student}} \quad \dfrac{\text{Upward-entailment for } s_{\text{AtLeast[5]}}}{A' \supseteq A \Rightarrow s_{\text{AtLeast[5]}}(A') \subseteq s_{\text{AtLeast[5]}}(A)}}{s_{\text{AtLeast[5]}}(\textbf{student}) \subseteq s_{\text{AtLeast[5]}}(\textbf{student} \cap \textbf{Japanese})} \quad \dfrac{\text{Premise}}{s_{\text{AtLeast[5]}}(\textbf{student} \cap \textbf{Japanese}) \subseteq D_3}}{s_{\text{AtLeast[5]}}(\textbf{student}) \subseteq D_3} \quad \dfrac{\text{Algebraic Property}}{D_3 \subseteq D_3'}}{s_{\text{AtLeast[5]}}(\textbf{student}) \subseteq D_3'}$$

Figure 4: An example of proof with generalized quantifiers encoded as selections.

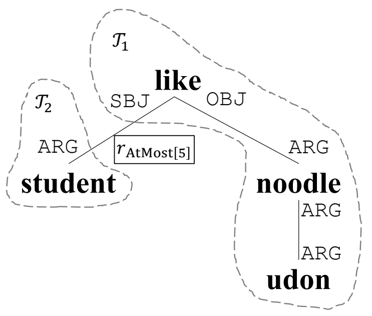


Figure 5: Encoding *"at most 5"* as relation

Intuitively, a GQ F can be represented by a relation r_F:

$$F(A)(B) \equiv r_F(A, B)$$

To enable the entailment from universal or to existential quantification, we simply add the axiom $A \subseteq B \Rightarrow r_F(A, B)$ or $r_F(A, B) \Rightarrow A \cap B \neq \emptyset$, respectively.

The axioms for monotonicity are also very intuitive. For GQs that are downward-entailing in both arguments (e.g. "*at most 5*" in Example 1), we put

$$r_f(A, B) \wedge A \supseteq A' \wedge B \supseteq B' \Rightarrow r_f(A', B').$$

Other kinds of monotonicity can be achieved in a similar way.

As for conservativity, we can simply implement

$$r_F(A, B) \Rightarrow r_F(A, A \cap B),$$

but the reverse

$$r_F(A, A \cap B) \Rightarrow r_F(A, B)$$

is a little tricky. This is because Tian et al. (2014a)'s inference engine is based on forward-chaining: it always tries to deduce *all* possible implications from given premises. This strategy is employed not only because of its efficiency, but also because it opens the possibility of adapting DCS for entailment generation (Androutsopoulos and Malakasiotis, 2010), in which case without any given hypotheses the system needs to actively explore what the premises entail. For example, from "few dinosaurs are pterosaurs", with the knowledge of GQ conservativity the system should figure out "few dinosaurs can fly" without being explicitly instructed to prove so. However, to implement $r_F(A, A \cap B) \Rightarrow r_F(A, B)$, the forward-chaining strategy would require the engine to find all Bs that satisfy $X = A \cap B$ whenever a relation $r_F(A, X)$ is claimed, in order to claim the relation $r_F(A, B)$. Though it is quite easy to check if $X = A \cap B$ for a given triple (X, A, B), issues arise when B is not given and we need to find all possibilities. It is generally impractical to enumerate all possible forms that a set X can be written as intersections; the number of possibilities easily explodes even for small-size problems[4]. Hence, we implement the rule $r_F(A, A \cap B) \Rightarrow r_F(A, B)$ as the following: if $r_F(A, X)$, *and* if $X \subseteq A$, then we take every $B \supseteq X$ and check if $X = A \cap B$. The necessary conditions $X \subseteq A$ and $B \supseteq X$ limit the search space at first. We would like to emphasize this detailed implementation issue here because formal semantics researchers are often not aware of such difficulties.

A shortcoming of the relation implementation is that, when processing DCS trees with relations, our extended system simply discards the edges marked as relations, then calculate the abstract denotations of germs in the resulting DCS forest, and finally use the denotations of corresponding germs as arguments of the relations to form statements. For example, in Figure 5, we calculate the denotations of germs $(\textbf{student}, \texttt{ARG})_{\mathcal{T}_2}$ and $(\textbf{like}, \texttt{SBJ})_{\mathcal{T}_1}$, which are **student** and D_3 (as defined in Section 2.2), respectively; then we form the statement $r_{\text{AtMost[5]}}(\textbf{student}, D_3)$ as the meaning of this sentence. This procedure implies that, relations in DCS trees are always explained as having the widest

[4]For example, $X = X \cap C$ for every $C \supseteq X$; even we only consider *minimal* intersections such that $X = A \cap B$ but $X \neq A$ and $X \neq B$, the possibilities could be exponential, e.g. consider $X = (A \cap B) \cap C = A \cap (B \cap C)$.

scope and hence we cannot deal with multiple relations in a single sentence. It causes errors when there are multiple GQs encoded as relations appear in the same sentence; we analyze this case in detail in Section 4.

4 Evaluation and Analysis

The FraCaS corpus (The Fracas Consortium et al., 1996)[5] was built in the mid 1990s by the FraCaS Consortium, which contains a set of hand-crafted entailment problems covering a wide range of semantic phenomena, organized in nine sections. The first section is titled "Generalized Quantifiers", and can serve as a good empirical test suite for RTE systems that handle general properties of GQs. This section contains 74 problems[6]; 44 of them have one premise sentence while the other 30 have multiple premises. The involved GQs and their properties are listed in Table 1.[7] Our implementation extends the TIFMO system publicly released with Tian et al. (2014a)[8]. Since we mainly focus on the performance of the DCS framework as formal semantics, on-the-fly knowledge and WordNet are not used. Major GQ properties can be implemented as composable and reusable units[9], so that each GQ can be created by simply composing the units that corresponds to the properties it has. This makes implementing new GQs very easy.

TIFMO uses the Stanford Parser[10] to obtain Stanford dependencies (de Marneffe et al., 2006) and POS tags, which are used to construct DCS trees based on a set of pre-defined rules. We extend those rules in order to recognize GQs in this step, and encode them under one of four settings, namely "Baseline", "Selection", "Relation", and "Selection+Relation". GQs are simply dropped in the "Baseline" setting. The "Selection" and "Relation" settings use the same DCS trees as in "Baseline", except for selection or relation markers on DCS trees to represent GQs. The "Selection" approach implements all GQs as selections (even for those are downward-entailing in the predicate argument), whereas "Relation" approach implements all GQs as relations. In the "Selection+Relation" setting, we use relations only for the GQs that are downward-entailing in the predicate argument (i.e. "few" and "at most n"), and implement the rest of the GQs as selections. We evaluate the system under each setting; the test results are shown in Table 2.

GQ	Entailed by $\forall$	Entails $\exists$	Monotonicity	
			Noun Arg.	Predicate Arg.
many	✓	✓	✗	$\uparrow$
a lot of	✓	✓	✗	$\uparrow$
few	✗	✓	$\downarrow$	$\downarrow$
a few	✓	✓	$\uparrow$	$\uparrow$
most	✓	✓	✗	$\uparrow$
at most n	✗	✗	$\downarrow$	$\downarrow$
at least n	✗	✓	$\uparrow$	$\uparrow$

Table 1: Properties of GQs appear in FraCaS corpus, including the interaction with universal and existential quantifications, and the monotonicity in noun and predicate arguments, where "$\uparrow$", "$\downarrow$", and "✗" denote upward-entailing, downward-entailing, and non-monotone, respectively.

System		Accuracy		
		Single	Multi	Overall
NatLog	MacCartney07	84.1%	N/A	
	MacCartney08	**97.7%**		
CCG-Dist	Parser Syntax	70.5%	50.0%	62.2%
	Gold Syntax	88.6%	80.0%	85.1%
TIFMO	Baseline	79.5%	86.7%	82.4%
	Selection	90.9%	93.3%	91.9%
	Relation	88.6%	93.3%	90.5%
	Selection+Relation	93.2%	**96.7%**	**94.6%**

Table 2: Accuracies achieved on the first section of FraCaS corpus using different systems.

We compare our results with two previous textual inference systems, CCG-Dist (Lewis and Steedman, 2013) and NatLog (MacCartney and Manning, 2007; MacCartney and Manning, 2008), also shown

[5]We used the version converted to XML format by MacCartney and Manning (2007).

[6]6 problems that do not have a defined solution are excluded.

[7]FraCaS dubiously interpreted "many" as denoting "a large proportion" rather than "a large absolute number", whereas "few" as denoting "a small absolute number" rather than "a small proportion". We also treat "a lot of" as a synonym of "many".

[8]`http://kmcs.nii.ac.jp/tifmo/`

[9]We implement GQ properties as stackable traits in Scala (Odersky et al., 2011), each consists of no more than a few dozen lines of code.

[10]`http://nlp.stanford.edu/software/lex-parser.shtml`

in Table 2. CCG-Dist uses rule-based conversion from CCG parses to first order logic formulas, and results are given using both parser syntax and gold syntax. The resulting accuracies are not very high, even for gold syntax, showing that implementing GQs is not an easily accomplishable task although first order logic is theoretically very expressive. NatLog is a system based on natural logic, which has almost perfect performance on single premise problems but faces difficulties dealing with premises of multiple sentences. In contrast, our extension of the TIFMO system achieves the best overall accuracy.

In each setting of our extension, almost all of the errors are related to the handling of GQs. The "Selection" approach cannot encode downward entailment in the predicate argument, as shown in Example 8; whereas the "Relation" approach fails to handle multiple GQs in a single sentence, as shown in Example 9.

Example 8. $P_1 \wedge P_2 \wedge P_3 \Rightarrow H$, where
P_1 Few committee members are from southern Europe.
P_2 All committee members are people.
P_3 All people who are from Portugal are from southern Europe.
H There are few committee members from Portugal.

Example 9. $P \not\Rightarrow H$, where
P At most ten commissioners spend a lot of time at home.
H At most ten commissioners spend time at home.

In Example 9, when both *"at most ten"* and *"a lot of"* are encoded as relations, both of them take the widest scope and the meaning of P is calculated as the conjunction of P_a *"at most ten commissioners spend (something) at home"*, and P_b *"(somebody) spend a lot of time at home"*. Then, P_a implies H since *"at most ten"* is downward-entailing in the predicate argument; the system produces a wrong answer. On the other hand, in the "Selection+Relation" setting, *"a lot of"* is encoded as selection and accompanied with the quantification marker "$\subseteq$", which can take a narrow scope and is explained as a division operator. Hence the calculated meaning of P becomes

$$r_{\text{AtMost[10]}}(\textbf{comm'r}, q_{\subseteq}^{\texttt{OBJ}}(D, s_{\text{ALotOf}}(\textbf{time})))$$

where D is the abstract denotation for *"spend at home"*

$$D = \textbf{spend} \cap (W_{\texttt{SBJ}} \times W_{\texttt{OBJ}} \times \textbf{home}_{\texttt{MOD}})$$

which is correct and solves this case. However, in general we need to further extend the notion of "relation" to handle different scopes, or at least we need something similar to the division operator but can be used to implement downward entailment in the predicate argument.

If we recall the definition of division operator $q_{\subseteq}$, it is natural to consider a similar operator as

$$q_{\supseteq}^{r}(R, C) = \{x \mid R \cap (\{x\} \times W_r) \supseteq \{x\} \times C_r\}\,.$$

Fortunately, $q_{\supseteq}$ can be defined algebraically as

$$q_{\supseteq}^{r}(R, C) = \pi^r(R) \setminus \pi^r(\bar{R} \cap (W^r \times C_r)),$$

where W^r denotes the Cartesian product of Ws on all dimensions other than r. This is implementable if we introduce complement $\bar{X}$ into Tian et al. (2014a)'s logical system. Operator "$q_{\supseteq}$" can be combined with selection operators to encode GQs that are downward-entailing in predicate argument, e.g. "at most ten". We may also be tempted to introduce free variables or higher order operators, especially when we begin to consider donkey anaphora (Heim, 1982) and other advanced inference phenomena. Such decisions should be made with caution because unguarded free variables easily lead to undecidability, as suggested by the research on description logics. However, further exploration on this topic should be a future direction but out of the scope of this work.

5 Conclusion

Encoding the semantics of a generalized quantifier is often crucial to correctly capturing the semantics of a sentence and making the right textual entailment. We have shown in this paper that major properties of GQs can be implemented in the DCS inference framework to correctly handle semantic relations like hyponymy. This tested and demonstrated the capabilities and potentials of the DCS framework, and suggested extensions towards more powerful logical systems for handling more sophisticated linguistic phenomena.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, 38(1):135–187, May.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Ronald J. Brachman. 1985. On the epistemological status of semantic networks. In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufmann.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS98)*.

Edgar F Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC 2006)*, pages 449–454.

Irene Heim. 1982. *The semantics of definite and indefinite noun phrases*. Ph.D. thesis.

Kevin Knight and Steve K. Luk. 1994. Building a large-scale knowledge base for machine translation. In *Proceedings of AAAI '94*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning Dependency-based Compositional Semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Stroudsburg, PA, USA. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Per Lindström. 1966. First order predicate logic with generalized quantifiers. *Theoria*, 32(3):186–195.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 521–528, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 41–48. Association for Computational Linguistics.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In Patrick Suppes, Julius Moravcsik, and Jaakko Hintikka, editors, *Approaches to Natural Language*, volume 49, pages 221–242. Dordrecht.

Andrzej Mostowski. 1957. On a generalization of quantifiers. *Fundamenta mathematicae*, 44:12–36.

Martin Odersky, Lex Spoon, and Bill Venners. 2011. Traits as stackable modifications. In *Programming in Scala: A Comprehensive Step-by-Step Guide*, chapter 12.5, pages 267–271. Artima Inc, 2nd edition.

John F. Sowa, editor. 1991. *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann.

The Fracas Consortium, Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. Using the framework. *Fracas project LRE 62*, 51.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014a. Logical inference on dependency-based compositional semantics. In *Proceedings of Association for Computational Linguistics (ACL) 2014*.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014b. Efficient logical inference for semantic processing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*.

Edward N. Zalta. 2014. Gottlob frege. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition.