A Modular Architecture for the Wide-Coverage Translation of Natural Language Texts into Predicate Logic Formulas

Yusuke Miyao[*] Alastair Butler[†‡] Kei Yoshimoto[‡] Jun'ichi Tsujii[§]

[*]National Institute of Informatics
[†]Japan Science and Technology Agency, PRESTO
[‡]Center for the Advancement of Higher Education, Tohoku University
[§]Graduate School of Information Science and Technology, Tokyo University

Abstract. We present a new method for translating unrestricted natural language texts into predicate logic formulas. This relies on the semantic evaluation procedure of Scope Control Theory (SCT), a variant of Dynamic Semantic formalisms. The key benefit is that parsed syntactic structures are shown to form sufficient input for semantic evaluation, eliminating the need to build distinct semantic expressions to feed semantic evaluation. To have parsed syntactic structures for SCT to evaluate we apply an existing wide-coverage syntactic parser by converting the parser output into a form SCT can receive. This modularity has led to the rapid attainment of a broad coverage on real text. An experiment revealed our system achieved 82.7% coverage on real-world sentences, generating representations that make explicit the scopes of quantifiers (e.g., $\exists x$), operators (e.g., negation), connectives (e.g., conjunction) and embedding predicates (e.g., *thinks*), while also capturing the inter and intra sentential dependencies and cross-sentential anaphoric dependencies that connect predicates.

Keywords: semantic evaluation, parser output, predicate logic formulas, unrestricted natural language

1 Introduction

Translating natural language expressions into logical formulas has been extensively studied for decades, attracting theoretical and practical interests. To date techniques have essentially involved adding methods of building a semantic expression piecewise to the process of identifying syntactic structure with a parser. In this paper we propose a departure from this tradition, with a system of semantic evaluation, Scope Control Theory or SCT (Butler, 2010), that accepts parsed syntactic structures as input. We employ semantic evaluation to generate explicitly semantic translations in the form of predicate logic formulas. It is also possible to have SCT evaluation generate the representations of alternative formalisms and offer different granularities for making semantic information explicit (e.g., generate encodings with more fine-grained presuppositional information, modal information, or tense information).

In using semantic evaluation to generate semantic translations a notable contribution is the elimination of the need for enriched guidance information that is extra to a conventional parsed form. Notably the co-indexing of syntactic constituents is rendered unnecessary. Moreover sentence information and discourse information are dealt with equally at the time of evaluation as providing binding information, which allows for their seamless integration.

The language formalism of SCT is supplemented by a compact grammar to determine the contribution of morphosyntactic information. This offers flexibility in what is acceptable as input, so there is no need to develop a SCT-specific parser from scratch. Here we adopt an HPSG-based syntactic parser (Miyao and Tsujii, 2008), because it offers wide coverage and high accuracy, and provides detailed syntactic information that is sufficient for SCT. Although the output of the parser is not directly compatible with the input assumed by SCT, a small conversion program can fill this gap, and consequently we achieve a wide-coverage system for translating unrestricted natural language texts into predicate logic formulas.

A notable advantage of the modular architecture is that system development is comparatively easy. As we prove in the experiments, a compact grammar of SCT and a small program for parsed syntactic structure conversion can achieve high coverage on real-world texts. This attests the portability of our method to other languages, as well as to other forms of semantic representation.

To our knowledge, Boxer (Bos *et al.*, 2004; Curran *et al.*, 2007) is the only alternative to our system that translates unrestricted texts into logical formulas (Discourse Representation Structures) with resolved intra and inter argument dependencies and anaphoric dependencies. We discuss the relationship with Boxer in Section 6 and note some added benefits of our approach.

2 Scope Control Theory

Scope Control Theory or SCT (Butler, 2010) is a system of semantic evaluation that attempts to approximate the dependency restrictions of natural language by a fine-grained and restricted management of binding dependencies, following insights from static reformulations of Dynamic Semantics (Cresswell, 2002; Dekker, 2002). Dynamic Semantics (Groenendijk and Stokhof, 1991) replicates the empirical results of Discourse Representation Theory (DRT) (Kamp and Reyle, 1993) independently of representation.

While SCT is a generic system for semantic evaluation that accepts syntactic structures as input, this paper specifically employs SCT to return semantic representations as output in the form of predicate logic formulas that make explicit the scopes of quantifiers (e.g., $\exists x$), operators (e.g., negation), connectives (e.g., conjunction) and embedding predicates (e.g., *thinks*), while also capturing the inter and intra sentential dependencies and cross-sentential anaphoric dependencies that connect predicates. For example, SCT is used to return the representation (2) for discourse (1).

(1) A man$_1$ smiled. He$_{1/*2}$ laughed. Another man$_2$ laughed with him$_{1/*2}$.

(2) $\exists xy(y \neq x \wedge \text{man}(y) \wedge \text{man}(x) \wedge \text{smile-past}(x) \wedge \exists z(z = x \wedge \text{laugh-past}(z)) \wedge \exists z(z = x \wedge \text{laugh-past +with}(y, z)))$

The pronouns of (1) contribute links with the form $\exists z(z = x \wedge ...)$ to the indefinite of the first sentence, while being prohibited from linking to the indefinite of the third sentence. Also *another* contributes the condition $y \neq x$. SCT controls such binding dependencies with *primitive operations* for altering the allocation of binding dependencies to binding names (Vermeulen, 2000). Binding names are labels to represent syntactic, semantic, or contextual roles (e.g., `"x"` for subject binding, `"y"` for object binding, and `"c"` for the source of antecedents for anaphora). We omit the detailed mechanism of SCT due to the space limitation. Refer to Butler (2010) for a complete description.

The notable feature of SCT we rely on in this work is that it accepts parsed syntactic structures as input for semantic evaluation. For example, for discourse (1) SCT requires the syntactic representation shown in Figure 1 as input. In this figure, `some`, `r`, and `pronoun` are grammar entries in SCT, which act as syntactic sugar for the composition of primitive operations. The operation `some` acts as a tigger for a superordinate existential closure operation to bring about the creation of a fresh binding, and `r` introduces a nominal or predicative relation where the first parameter (e.g. `["h"]` and `["x"]`) expresses the required argument or arguments of the relation that must be bound, the second parameter (e.g. `["with"]`) corresponds to arguments for any attachments that are usually bound by preposition phrases, and the third parameter denotes long-distance dependencies (described in Section 4). The operation `pronoun` plays a crucial role in creating anaphoric dependencies; it picks up an accessible antecedent from a sequence ranked by salience, and creates a binding dependency.

In addition, there are a few operators that combine two SCT expressions, notably $\backslash\backslash$ which behaves like the backslash of categorial grammar to control the ordering of an instance of functional

```
(some [r ["h"] [] [] "man"]  ⑊ r ["x"] [] [] "smile-past")
(pronoun  ⑊ r ["x"] [] [] "laugh-past")
(some [r ["h"] [] [] "man"]  ⑊ (r ["x"] ["with"] [] "laugh-past+with"  ⫽ pronoun))
```

Figure 1: SCT input for the discourse (1)

application, while also providing the "x" (subject) binding name information; and ⫽ which behaves like the forward slash of categorial grammar, while also providing the "y" (object) binding name information.

Syntactic structures represented with these grammar entries and combination operators are reduced into primitive operations, on which SCT performs semantic evaluation.

3 A syntactic parser: Enju

As we can see from Figure 1, SCT requires the following syntactic information as input: (A) subcategorisation frames of words, i.e., how many and what phrase categories each word takes as arguments; (B) constituent structures, including construction types such as the subject-verb construction; and (C) long distance dependencies, i.e., what arguments in an embedded clause have to be passed up. Most syntactic parsers that are widely used recently, such as treebank parsers (Charniak and Johnson, 2005) and shallow dependency parsers (Nivre and Nilsson, 2005; McDonald and Pereira, 2006), cannot directly provide the above information. How to feed sufficient syntactic information is the key issue for the SCT evaluation to work.

In order to obtain this information, we adopt *Enju* (Miyao and Tsujii, 2008), an English syntactic parser based on HPSG (Pollard and Sag, 1994; Sag *et al.*, 2003). HPSG is a lexicalist framework of syntactic theory; that is, various syntactic constructions are explained with lexical entries. Refer to Pollard and Sag (1994) and Sag *et al.* (2003) for the details of HPSG.

The Enju parser has several advantages to achieve our goal. One is that, owing to the lexicalist framework, the grammar encodes most syntactic information, including subcategorisation frames, in lexical entries. This can be used directly to obtain the information (A). Furthermore, constituent structures output by the Enju parser provide not only phrasal categories, but also HPSG-style construction types, such as subject-head and relative clause constructions, which can be mapped to information (B). In addition, HPSG explains long-distance dependency in a lexicalised way; it explicitly encodes how a trace percolates from a lexical entry to the place where the moved argument is filled. This is useful for computing the information (C). Lastly, this parser is robust; it can produce a complete parsed form for 99.7% of news-wire texts (Miyao and Tsujii, 2008).

Although the Enju parser works internally with HPSG-based syntactic structures, it can output syntactic information in a theory-independent representation in an XML format (Miyao, 2008). Rather than the full HPSG-style representation, the Enju XML format is easier to use, and includes sufficient information to obtain the syntactic information that SCT requires. We therefore start from the XML output of Enju for obtaining the input to the SCT evaluation.

Figure 2 shows the XML output for the simple sentence (3), while Figure 3 illustrates the same information in a readable form, where the value of `cat` is represented as a phrase label and the values of selected attributes are put as subscripts.

(3) A man smiled.

In the Enju XML format, the tag `cons` represents constituents, and `tok` denotes words. These tags have attributes to express syntactic/semantic information. The essential attributes used for constructing the input to SCT are listed below.

`cat` phrasal category label

`xcat` labels to express additional syntactic features: e.g. `TRACE` for constituents having a trace, and `REL` for relative pronouns/clauses

```
<cons id="c0" cat="S" xcat="" schema="subj_head">
 <cons id="c1" cat="NP" xcat="" schema="spec_head">
  <cons id="c2" cat="DP" xcat="">
   <tok id="t0" cat="D" pred="det_arg1" arg1="c3">A</tok>
  </cons>
  <cons id="c3" cat="NX" xcat="">
   <tok id="t1" cat="N" pred="noun_arg0">man</tok>
  </cons>
 </cons>
 <cons id="c4" cat="VP" xcat="">
  <tok id="t2" cat="V" pred="verb_arg1" tense="past" aspect=
   "none" voice="active" aux="minus" arg1="c1">smiled</tok>
 </cons>
</cons>
```

Figure 2: An XML output from Enju

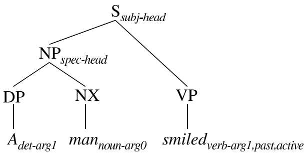

Figure 3: An XML output from Enju, in a graphical representation

`schema` HPSG-style construction type

`pred` predicate type, i.e., subcategorisation frame

Refer to Miyao (2008) for the complete description of the Enju XML format. Although the Enju parser outputs predicate argument dependencies with the attributes `arg1, ..., arg4`, they are used only for identifying the phrase categories of arguments, and not explicitly used. Semantic dependencies, including predicate argument dependencies, are computed by SCT solely.

4 Conversion from Enju XML to SCT

Consider (4) which can be encoded as in Figure 4, and its evaluation results in the predicate logic formula of (5).

(4) A man spoke to a man who he thought was smiling.

(5) $\exists xy(\text{man}(x) \wedge \exists z(z = y \wedge \text{think-past}(z, \text{smile-prog-past}(x))) \wedge \text{man}(y) \wedge$
$\text{speak-past +to}(y, x))$

```
some [r ["h"] [] [] "man"]  \\
 (r ["x"] ["to"] [] "speak-past"  //
  (prep "to" (some [r ["h"] [] [] "man", relc "x" wh_phrase
   (pronoun  \\ (remb ["x"] [] ["x"] "think-past"  // (comp (r ["x"] [] [] "smile-prog-past"))))]])))
```

Figure 4: Input to SCT for sentence (4)

Figure 5 shows the Enju XML output for (4) which is largely isomorphic to its SCT counterpart. We need to establish a method for converting Figure 5 into Figure 4. The core part of our conversion procedure is summarised in the following two steps: (1) convert terminal nodes of Enju output into lexical entries of the SCT grammar, and (2) convert constituent structures of Enju into (sub)expressions of the SCT grammar.

Step (1) is achieved by one-to-one mapping rules for terminal nodes, i.e., `tok`. For open-class words, including nouns, verbs, etc., mapping rules refer to the predicate type, i.e., `pred`, of a word, and additionally to `cat` of arguments to identify phrasal categories of arguments. For closed class words, i.e., determiners, conjunctions, etc., mapping rules are dependent on word forms (Table 1). We have implemented 75 conversion rules for terminal node mappings.

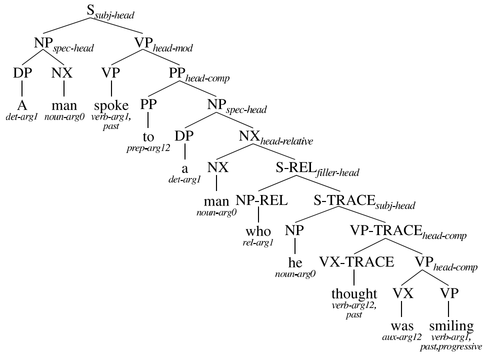

Figure 5: The Enju XML output for sentence (4)

word	SCT grammar entry
a	`some`
the	`the`
he	`pronoun`
who	`wh_phrase`

Table 1: Mapping of terminal nodes for closed-class words

Step (2) involves traversing a tree structure of XML and producing a SCT expression for each internal node. This conversion is accomplished by referring mainly to `schema`, which denotes HPSG-style construction types, and optionally `cat` and `xcat` of child nodes. Table 2 shows a part of such mapping rules, where *`L`* and *`R`* indicate SCT expressions of left and right child nodes, and x denotes a binding name. The conversion rules for constituent structures have been implemented for 52 construction types.

While the major part of the format conversion consists of straightforward mapping rules as described above, there are non-trivial differences in the specifications of the Enju XML and the input SCT assumes. Major differences are in the treatment of modifiers and long-distance dependencies. Thus, we need the preprocessing of the XML output for these constructions.

Enju regards modifiers (adjectives, prepositional phrases, etc.) as attaching to what they modify; on the other hand, for the input to SCT it is supposed that modifier information is encoded in the grammar entry of the head word. For example, the grammar entry for *spoke* in Figure 4 includes a binding name `"to"` in its second parameter, which indicates that *spoke* is modified by a phrase with the binding name `"to"`. Since the Enju counterpart does not explicitly have information about its modifiers, we collect all modifiers for each word prior to format conversion, and use this information to fill out attachment binding names in grammar entries.

Long-distance dependencies are handled in a similar way. The SCT input requires that binding names for moved arguments have to be specified explicitly for each predicate they pass over. For example, the third parameter of *thought* in Figure 4, `["x"]`, means that the `"x"` binding at this point (which comes from *smile*) is passed over this verb. Since `tok` for *thought* does not have this information, we preprocess Enju output to collect constituents that have `xcat="TRACE"` (which indicates the constituent has a trace), and register passing-over binding names for the semantic head of each of these constituents. Registered binding names are used to fill out `ext` in grammar entries. A similar procedure is applied for relativised arguments, by referring to `xcat="REL"`.

construction type	SCT expression
subject-verb construction (schema="subj_head")	$L \quad \backslash\backslash\ R$
verb-complement construction (schema="head_comp")	$L \quad //\ R$
specifier-noun construction (schema="spec_head")	$L\ [R]$
relative clause construction (schema="head_relative")	L, (relc x R)

Table 2: Mapping of constituent structures

Stately, plump Buck Mulligan came from the stairhead, bearing a bowl of lather on which a mirror and a razor lay across. A yellow dressinggown ungirdled was sustained gently behind him on the mild morning air.

$\exists xyzuvwx_1$(mild_morning_air(w) $\wedge$ $\forall x_2$(mild_morning_air(x_2) $\rightarrow$ $x_2 = w$) $\wedge$ yellow_dressinggown_ungirdled(x_1) $\wedge$ stairhead(v) $\wedge$ $\forall x_2$(stairhead(x_2) $\rightarrow$ $x_2 = v$) $\wedge$ razor(z) $\wedge$ mirror(y) $\wedge$ lather(x) $\wedge$ bowl +of(u, x) $\wedge$ lay_across +on(y, u) $\wedge$ lay_across +on(z, u) $\wedge$ bear-prog(plump_buck_mulligan, u) $\wedge$ stately_come-past +from(plump_buck_mulligan, v) $\wedge$ $\exists x_2$(x_2 = plump_buck_mulligan $\wedge$ sustain-past-passive_gently +behind +on(x_1, x_2, w)))

Figure 6: Example output

5 Examples and experiments

We have implemented the SCT grammar for English and the conversion program from Enju XML into the SCT input form, by referring to Penn Treebank Section 22 (Marcus *et al.*, 1994), which is a portion often used for parser development. Table 3 shows the code size of the implemented programs (including comments and blank lines) at this moment. This system can produce complete predicate logic formulas for 84.2% of the sentences of this development set. The principal reason for the remaining uncovered sentences was the lack of SCT grammar entries and/or conversion rules, for constructions including some types of coordination, *tough* construction, etc. Since the extension of our system to these constructions does not involve any theoretical/practical difficulties, further improvement in coverage is expected in future work. Another reason was linguistically illformed syntactic structures output by the parser, and for reducing such errors the improvement of the parser is demanded.

Figure 6 shows an example output from our system for the opening lines of *Ulysses* by James Joyce. This demonstrates various kinds of dependencies, including the participial construction, the formation of a relative clause containing a subject involving the coordination of indefinites, uniqueness requirements from the definite articles, as well as a pronoun appropriately linked to its antecedent, all of which is not explicitly represented in the syntactic structure input and therefore computed by SCT.

Finally, we evaluated the robustness of our system on unseen real-world texts. We used Section 00 of the Penn Treebank as a test set. We excluded the data that are nonsentences such as titles (79 lines), and for which the Enju parser did not output the complete analysis (6 sentences); consequently the test set consists of 1836 sentences. These sentences were not seen for the development of our system.

Our system produced complete well-formed predicate logic formulas for 1518 sentences, showing 82.7% coverage. It is striking to see this practical coverage on real-world texts, considering the strict management of binding dependencies and scopes, as well as the conciseness of the programs we implemented. It should also be noted that the coverage figure for the test set was not

Program	Lang.	# lines
SCT evaluation	ML	643
SCT grammar for English	ML	403
Conversion from Enju XML to SCT	Perl	1212

Table 3: Code size of the implemented programs

significantly different from the figure for the development set, indicating that our system is not over-tuned to the development set. We suppose this is because of the generality of the implemented grammar and the conversion rules.

Currently we are unable to empirically measure the accuracy of the resulting semantic representations due to the cost for the manual construction of gold standard semantic representation data. This is a limitation we expect to resolve in future work by constructing a corpus of semantic representations.

6 Related work

Boxer (Bos *et al.*, 2004; Curran *et al.*, 2007) is a system that achieves wide-coverage translation of natural language texts into the representations of Discourse Representation Theory (DRT). It receives CCG derivations from the C&C parser (Clark and Curran, 2004), and maps lexical categories and combinatory rules into semantic descriptions expressed by lambda calculus. In our SCT based approach dependencies are established by a runtime evaluation rather than by representation, which has as a notable advantage the property of automatically tracking both accessibility and salience.

Boxer was claimed to be modular because one can alter the semantic representation by changing the syntax-to-semantics mappings, but this would involve a reimplementation of the whole semantic part. Our system is more modular; because the syntactic form received by SCT is fixed, to switch output representation we need only adjust the evaluation procedure. This leaves much room for experimenting with non-standard forms of evaluation that might for example lead to evaluations where potentially ambiguous relations and binding dependencies are kept underspecified.

7 Conclusion

The SCT system is most interesting because it offers a semantic evaluation procedure that accepts parsed syntactic structures of natural language as evaluable expressions. This is achieved with the aid of a concise grammar that allows parsed morphosyntactic information to be treated as syntactic sugar for primitive operations over assignment functions. With the help of a wide-coverage syntactic parser, we have been able to develop a robust system for semantic processing at relatively low cost. The experiment showed our current system achieves 82.7% coverage on Penn Treebank sentences.

References

Bos, Johan, Stephen Clark, and Mark Steedman. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING 2004*.

Butler, Alastair. 2010. Semantically restricted argument dependencies. *Journal of Logic, Language and Information*, DOI: 10.1007/s10849-010-9123-8.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL 2005*.

Clark, Stephen and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of ACL 2004*.

Cresswell, M. J. 2002. Static semantics for dynamic discourse. *Linguistics and Philosophy*, 25, 545–571.

Curran, James R., Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pp. 33–36.

Dekker, Paul. 2002. Meaning and use of indefinite expressions. *Journal of Logic, Language and Information*, 11, 141–194.

Groenendijk, Jeroen and Martin Stokhof. 1991. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14(1), 39–100.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 313–330.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*.

Miyao, Yusuke. 2008. Enju 2.3 output specifications. `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/enju-manual/enju-output-spec.html`.

Miyao, Yusuke and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1), 35–80.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. CSLI Publications.

Vermeulen, C. F. M. 2000. Variables as stacks: A case study in dynamic model theory. *Journal of Logic, Language and Information*, 9, 143–167.