Linguistic **I**ssues in **L**anguage **T**echnology – **LiLT**
Submitted, January 2012

Banking Meaning Representations from Treebanks

Alastair Butler
Kei Yoshimoto

Published by CSLI Publications

LiLT volume 7, issue 6 January 2012

Banking Meaning Representations from Treebanks

Alastair Butler, *Japan Science and Technology Agency, PRESTO,* Kei Yoshimoto, *Tohoku University*

Abstract

This paper describes a method to convert existing treebanks with syntactic information into banks of meaning representations. The central component is a system of evaluation for a small formal language with respect to an information state. Inputs to the evaluation system are formal language expressions obtained from the conversion of parsed representations conforming to (Penn Treebank Project) guidelines. Outputs from the evaluation system are Davidsonian (higher-order) predicate logic meaning representations. Having a system of evaluation as the basis for generating meaning representations makes possible accepting input with minimal conversion from existing treebanks and from the tools used to construct treebanks. Results of having built corresponding banks of meaning representations from available treebanks are discussed.

LiLT Volume 7, Issue 6, January 2012.
Banking Meaning Representations from Treebanks.

1 Introduction

This paper introduces a method of formal evaluation as a way to automatically obtain banks of meaning representations from data following conventional treebank guidelines. Applicability is demonstrated with data of The Penn Treebank Project (Marcus et al. 1993). The approach requires no lexical information that is additional to what is retrievable from the input treebank data, having the ability to build or adjust the contribution of morphosyntax information depending on a changing information state in which sequence values are assigned to binding names with grammatical roles. Binding name roles are prescribed on a language and treebank guidelines basis, supplemented with roles generated from a scan of the input data. The mechanism of the formal evaluation is of crucial importance because it acts to enforce guidance on the creation of binding dependencies, guaranteeing grammatical dependencies.

The paper is structured as follows. Section 2 outlines theoretical background for the approach. Section 3 sketches a simple example. Section 4 covers definitions on which the approach relies. Section 5 details the calculation of scope for quantifiers, indefinites and definites, including interaction with negation. Section 6 presents results for the conversion of existing treebanks into banks of meaning representations. Section 7 discusses related work. Section 8 summarises.

2 Theoretical Background

In the formal study of language the essence of grammatical structure is regarded as the range of valid dependencies. Linguistic theories typically work by placing limits on the manipulation of syntactic structures, e.g., constituent trees or feature structures, to attain a handle on valid dependencies (see for example Government and Binding Theory, Lexical Functional Grammar, Head-driven Phrase Structure Grammar, Combinatory Categorial Grammar and Dynamic Syntax).

The theory adopted here, Scope Control Theory (SCT; Butler 2010), takes a different route, aiming to characterise grammatical structure in terms of dependencies established when there is the formal evaluation of a given parsed expression against an information state as introduced by Vermeulen (2000) that assigns to binding names sequences of values storing discourse and intra sentential information. The ultimate purpose for undertaking evaluations is to calculate denotations of natural language utterances—here as meaning representation output.

For SCT well-formedness of a parsed expression is assured when it is possible to maintain during evaluation equality between the length

of sequence values required to satisfy the bindings of the parsed expression under evaluation and the length of sequence values provided by the current information state. By extending, manipulating, reducing or temporarily making inactive parts of assigned sequence values under this equality condition, the complex formation of natural language utterances is simulated.

The current work builds on Butler (2010) where it was shown how limits from evaluation can match a wide range of valid dependency patterns, including locality effects, accessibility of anaphoric referents, intervention effects and circumstances for long-distance dependencies. With the application of this theory the constraints from evaluation are themselves employed to assist evaluation when structural information is missing from a parsed input. Consequences are twofold: when structural information is present restrictions of grammaticality are enforced, when structural information is absent decisions about valid dependencies may guide evaluation.

3 An Example

To have evaluation generate meaning representations starts with data in some treebank format. For example, sentence (1) is represented as the parsed form (2) following the Penn Treebank II guidelines of Santorini (1990) and Bies et al. (1995).

(1) Pierre joins the board.

(2)
```
(S (NP-SBJ (NNP Pierre))
   (VP (VBZ joins) (NP (DT the) (NN board))) (. .))
```

The next step is to convert (2) into an evaluable expression, (3). This transforms into operators the part of speech tags inserted as nodes immediately dominating the terminals of (2). While `NP-SBJ` converts to the `"arg0"` binding name, and bare `NP` to `"arg1"`, other non-terminal nodes of (2) are eliminated to leave only the bracketed constituency of (2). Representation (3) also adds information about the range of potential binding names, with `["h", "arg0", "arg1"]` as the value for the `lc` parameter (the `"h"` name is always present and will bind nominals; see `"board"` in (5)).

(3)
```
( λlc.
  ( ( ( nnp "Pierre") "arg0")
    ( ( ( dt lc fh "the" "d" ( nn lc "board")) "arg1")
      ( present ( verb lc ["arg1"] "joins")))))
["h", "arg0", "arg1"]
```

What is interesting about (3) is that only limited information recoverable from (2) is given about the lexical content of the verb. The

conversion of `VBZ` leads to the verb being marked with `present` (tense), and sister information from (2) gives `["arg1"]` to state the verb has an object, but there is no other argument structure information to distinguish whether the verb takes a subject binding, or additional modifying arguments, etc.

Definitions (see section 4 for details) are given for `nnp` (proper name), `dt` (determiner), `nn` (ordinary noun), `present` and `verb`, such that, with the part of speech and functional tag information of the input tree, there is no requirement for a lexicon to state the contribution of specific words. (To manipulate the quality of meaning representations particular word information, especially for functional words, as well as details of specific analyses of linguistic phenomena, are typically integrated during conversion to an evaluable expression, e.g., the step from (2) to (3). This is deliberately left as an open ended task to foster experimentation.) Together with specification of the open parameter `fh`, e.g., `fh = ["e", "event", "d", "constant"]`, (3) reduces to the evaluable (4).

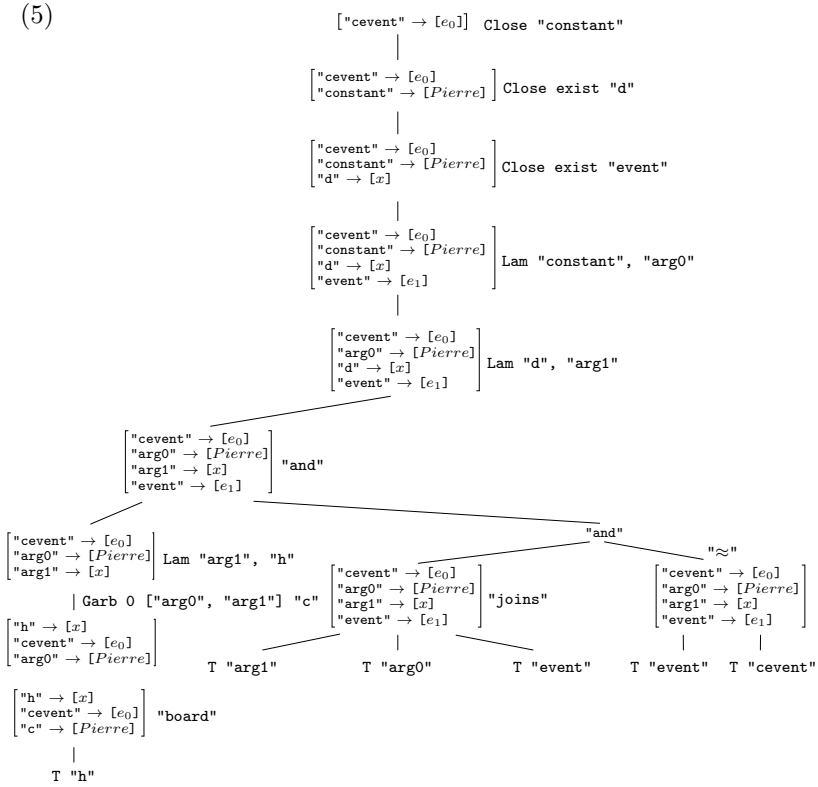
(4)
```
CUse ("Pierre", "constant",
 Lam ("constant", "arg0",
  Use ("d",
   Lam ("d", "arg1",
    Rel (["e", "event", "d", "constant"],
     ["c", "c", "c", "c"], "and", [
     Throw ("d",
      Lam ("arg1", "h",
       Garb (0, ["arg0", "arg1"], "c",
        Garb (1, ["h"], "c",
         Pred ("board" [T "h"])))))),
     Rel (nil, nil, "and", [
      Use ("event", [
       If (gt1test "h",
        If (gt1test "arg0",
         Pred ("joins", [T "arg1", "arg0", "h", "event"]),
         Pred ("joins", [T "arg1", "h", "event"])),
        If (gt1test "arg0",
         Pred ("joins", [T "arg1", "arg0", "event"]),
         Pred ("joins", [T "arg1", "event"])))),
      Throw ("event",
       Pred ("≈", [T "event, T "cevent"])))))))
```

Expression (4) is built from primitive operations of the SCT language (see Butler 2010 for details): `CUse` (states injection of a constant into the assignment), `Use` (states a binding must be available for a binding name), `Lam` (shifts bindings between binding names), `Rel` (creates a relation while manipulating the information state), `Pred` (creates a predicate), `T` (creates a bound term), `Throw` (instructs to reposition material with respect to a closure post-evaluation), `Garb` (removes active

bindings) and `If` (evaluates only one of two subexpressions depending on a test of the information state, e.g., `gt1test "h"` tests whether the sequence assigned to `"h"` contains one or more values).

A (simplified) illustration of what occurs with an evaluation of (4) is given by (5). Assume names missing from illustrations of assignments are assigned the empty sequence.

(5)



This starts from an initial information state where only `"cevent"` (context event, beginning as utterance time) is assigned a non-empty sequence, which contains a single value. Values are subsequently added to the sequences assigned to `"constant"`, `"d"` and `"event"`. The distribution of these values is further manipulated throughout (5), with sequence values able to shift between the sequences assigned to different binding names, so whenever evaluation terminates only correct bindings for the relevant predicates obtain. When a predicate leaves open a range of argument options (as with `"joins"` in (4), depending on an `If` condition testing for `"arg0"` bindings embedded in an `If` condition testing

for "h" bindings), the appropriate option is selected by the evaluation ((5) shows "joins" with "arg0" and "arg1" arguments and no "h" argument, and so as a transitive verb). If a predicate is fully specificed (as "board" is in (4) with no If conditions), error feedback about the information state is returned should the information state fail to satisfy what is required by the predicate.

Following (5) the meaning representation (6) is returned. This assumes a Davidsonian theory (Davidson 1967) in which the verb expresses a three-place relation between the nominal arguments and an implicit event argument which is existentially quantified over. Such a representation allows for any modifiers of the verb to be added conjunctively as predicates of the event argument. We should expect the free variable e_0 to denote the time of utterance as part of the encoding of present tense.

(6) $\exists x(\text{board}(x) \wedge \exists e_1(e_1 \approx e_0 \wedge \text{joins}(e_1,\text{Pierre},x)))$

4 Definitions

In this section we fill in details for nnp (proper name), present (tense), nn (ordinary noun), verb and dt (determiner), sufficient to reduce (3) to (4).

Proper names are formed with nnp, (7), invoking SCT primitives CUse to inject a constant (the value r) into the assignment with (i) a commanding "constant" closure that searches for correspondingly named instances of CUse, and (ii) Lam to make r a sequence value assigned to x. The point of utilising a commanding closure rather than more directly integrating the constant with an x binding is that availability of the constant is made as wide as the level of closure (e.g., to happen at the level of discourse). The contribution of the proper name is thereby made accessible for pronouns (but essentially only pronouns that follow the proper name within the text, because of Rel discussed at the end of section 4).

(7)
```
nnp = λr.λx.λemb.
 CUse (r, "constant", Lam ("constant", x, emb))
```

present, (8), creates a restrictive condition to state that the event time of the local verb (stored as the "event" binding) 'is contemporaneous with' (≈) the frontmost "cevent" binding (the utterance time with a matrix clause). Internally to present, Throw "event" provides an instruction to reposition after evaluation completes the restrictive condition immediately below the point where the event binding is created.

(8)
```
present = λemb.
 Rel (nil, nil, "and", [
```

```
emb,
Throw ("event",
  Pred ("≈" [T "event", T "cevent"])))
```

For `nn` and `verb`, we start by introducing recursive `add_args`, (9). This takes three parameters: `l1` and `l2` as sequences of binding names and `predicate` that itself has an open parameter to take a sequence of binding names. If `l1` is `nil` (the empty sequence) then the content of `l2` is applied to `predicate`, else an expression is created with `If` that has (i) a test for a single binding with `(hd l1)` (the frontmost value of `l1`), (ii) an expression to evaluate if the test succeeds involving `l2` extended by concatenation (`@`) with `[hd l1]` and a recursive call to `add_args` on the tail of `l1`, and (iii) an expression to evaluate if the test fails involving a recursive call to `add_args` on the tail of `l1` but no extension to `l2`.

(9)
```
rec add_args = λl1.λl2.λpredicate.
 if l1 = nil then predicate l2 else
   If (gt1test (hd l1),
     add_args (tl l1) ([hd l1] @ l2) predicate,
     add_args (tl l1) l2 predicate)
```

To create encodings for nouns `nn`, (10), calls `add_args` as a wrapper around `Pred` that creates a predicate with at the very least a `T "h"` argument, and possibly others built from names supplied to an `l` parameter. The call of `add_args` adds arguments to `Pred` with names taken from the open `lc` parameter, but with `diff` acting to remove the `"h"`, `"arg0"` and `"arg1"` names, such that added arguments only have consequences for an evaluation when sufficient binding support is present from the assignment.

(10)
```
nn = λlc.λs.
 add_args (diff (lc, ["h", "arg0", "arg1"])) nil (
   λl.Pred (s, map (λx.T x) (l @ ["h"])))
```

Encodings for verbs are created with `verb`, (11). This is similar to `nn`, except the ever present argument is `"event"` rather than `"h"`, and a `Use` instruction is given for an `"event"` binding to be created by a commanding level of closure. Also there is an extra parameter `args` for taking a sequence of binding names that must be arguments of the verb.

(11)
```
verb = λlc.λargs.λs.
 Use ("event",
  add_args (diff (l, args @ ["event"])) nil (
   λl.Pred (s, map (λx.T x) (l @ args @ ["event"])))))
```

Next we consider the definition of `dt` (determiner):

(12)
```
dt = λlc.λfh.λdet.λv.λl.λx.λemb.
 (case det
    of "a" => some lc fh v l x emb
     | "the" => some lc fh v l x emb
     | "every" => every lc fh v l x emb
     | _ => general_quant lc fh v l x emb)
```

Depending on the `det` parameter, `dt` calls either `some`, (13), `every`, (14), or `general_quant`, (15).

(13)
```
some = λlc.λfh.λv.λe.λx.λemb.
 Use (v,
  Lam (v, x,
   Rel (fh, map (λx."c") fh, "and", [
    Throw (v, rest lc x e), emb]))
```

(14)
```
every = λlc.λfh.λv.λe.λx.λemb.
 Hide(v,
  Close (∀, v,
   Use (v,
    Lam (v, x,
     Rel (fh, map (λx."c") fh, "→", [
      rest lc x e, Hide (v, Close (∃, v, emb)])))
```

(15)
```
general_quant = λoper.λlc.λfh.λv.λe.λx.λemb.
 Quant (oper, x,
  Rel (fh, map (λx."c") fh, "", [rest lc x e, emb]))
```

`some` contains `Use v` requesting the commanding `v` closure to create a value which `Lam` integrates as a sequence value assigned to `x`. As with `nnp`, the rational for the `Use-Lam` combination is to produce discourse consequences: the introduced value is made accessible to subsequent pronouns. Presence of `Throw v` ensures material of the restriction has placement immediately below the point where the introduced value is created after evaluation completes.

`every` introduces operations of closure with the SCT primitive `Close`, to create overall instances of universal quantification based on a count for occurrences of `Use v`, as well as existential quantification internal to the nuclear scope. `Hide v` is a primitive SCT operation that terminates counting instances of `Use v` and `CUse v`. In contrast, the quantification of `general_quant` is created with the SCT primitive `Quant` which forms a single quantificational instance without closure/discourse consequences.

Correct environments for the restrictions of noun phrases are assured with the calls to `rest` included with (13)–(15). The idea implemented by `rest`, (16), is that noun phrases should insulate the restriction from the containing clause by shifting all open local bindings to a given

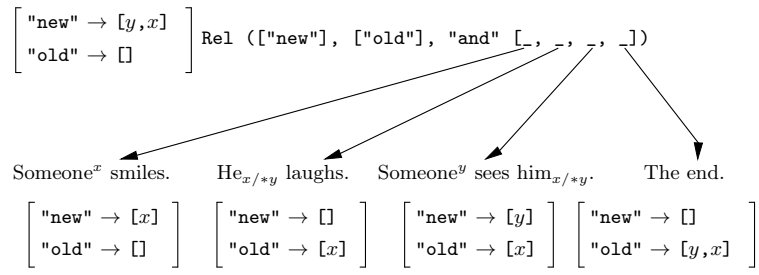
binding name (here, `"c"`) with the exception of the binding that it is the purpose of the noun phrase to introduce which must shift to an `"h"` binding to make available the required binding for nouns. Insulation is accomplished with SCT primitive operations `Lam` and `Garb`. `Lam` relocates the `x` binding (the binding the noun phrase opens in the containing clause) to an `"h"` binding. `Garb (`n`,` $\hat{x}$`,` y`,` e`)` modifies the assignment by relocating potentially multiple values from the sequences assigned to the names of $\hat{x}$ into the sequence that is assigned to y, so exactly the frontmost n bindings remain assigned to each name of $\hat{x}$.

(16)
```
rest = λlc.λx.λe.
 Lam (x, "h",
   Garb (0, diff (lc, ["h"]), "c",
     Garb (1, ["h"], "c", e)))
```

An application of `rest` can be seen within (5) that (i) turns the `"arg1"` binding into an `"h"` binding, (ii) shifts all other open bindings of the names in `["h", "arg0", "arg1"]` minus `"h"` to `"c"` bindings, and (iii) would shift all values of the sequence assigned to `"h"` to `"c"` with the exception of the frontmost value were there any, which is not the case in (5).

The remaining SCT primitive operation present in (13)–(15) to detail is `Rel`. The role of `Rel` is to establish a relation over a sequence of arguments, but in doing so `Rel` can also manipulate the content of the assignment function passed on for the evaluation of each argument to achieve discourse consequences based on the content of two sequences of binding names and the number of instances of `Use` present in the arguments with names corresponding to names in the first sequence. This is demonstrated with (17): `Rel` creates an `"and"` relation with four arguments, each of which are to be evaluated against a different state of the assignment function determined by instances of `Use "new"` which are present with the occurrences of *Someone* (e.g., following (13)). The idea reflected in (17) is that pronouns are able to link to `"old"` bindings.

(17)



5 Quantifiers, Indefinites and Definites

Crucial information required for meaning representations and yet missing from syntactic representations concerns the placement of scope taking elements together with their restriction materials. In this section we detail how this missing information is calculated during evaluations of quantifiers, indefinites and definites.

As a default quantifiers will take scope in the meaning representation in correspondence with their syntactic location within the input parse tree. For example, in (18) "every day" falls under the syntactic scope of "someone" and so gives rise to (19) in which the contribution of "every day" scopes under the contribution of "someone". Note "every day" existentially closes its nuclear scope to scope over the binding of the temporal argument of the verb. The temporal modifier representation in (18) is based on a suggestion in Landman (2000), using $\sqsubseteq$ for temporal inclusion.

(18)
```
(S (NP-SBJ (NN Someone))
   (VP (VBD visited)
       (NP (NNP Bill))
       (NP-TMP (DT every) (NN day)) (. .))
```

(19) $\exists x(\text{person}(x) \wedge \forall y(\text{day}(y) \rightarrow \exists e_1(e_1 < e_0 \wedge \text{visited}(e_1, x, \text{Bill}) \wedge \text{time}(e_1) \sqsubseteq y)))$

The scope of an indefinite like "someone" is not determined by its syntactic location, but rather by the placement of the closure from where its binding comes. In converting (20) to SCT expression (21) the contribution of "someone" is marked to take its binding from an `"e"` closure. With its nuclear scope "every day" brings about an `"e"` (existential) closure (as seen with the event binding in (19)), and so the returned meaning representation, (22), has "someone" scoping below "every day".

(20)
```
(S (NP-TMP (DT Every) (NN day))
   (NP-SBJ (NN someone))
   (VP (VBD visited)
       (NP (NNP Bill))) (. .))
```

(21)
```
( λlc.
  ( ( ( dt lc fh "every" "e" ( nn lc "day"))
      "tmp_")
    ( ( ( some lc fh "e" ( nn lc "person"))
        "arg0")
      ( ( ( nnp "Bill")
          "arg1")
        ( past ( verb lc ["arg1"] "visited"))))))
["h", "arg0", "arg1", "tmp_"]
```

(22) $\forall x(\text{day}(x) \rightarrow \exists e_1(e_1 < e_0 \wedge \exists y(\text{person}(y) \wedge \text{visited}(e_1, y, \text{Bill}) \wedge \text{time}(e_1) \sqsubseteq x)))$

In contrast, the contribution of "someone" in (23) is marked to take its binding from a `"d"` closure, which is the only deviation from (21). A `"d"` closure occurs only at the discourse level to give the indefinite widest scope, and so brings about the same output form the evaluation of (23) as seen already with (18), namely (19).

(23)
```
( λlc.
  ( ( ( dt lc fh "every" "e" ( nn lc "day")) "tmp_")
    ( ( ( some lc fh "d" ( nn lc "person")) "arg0")
      ( ( ( nnp "Bill") "arg1")
        ( past ( verb lc ["arg1"] "visited")))))))
["h", "arg0", "arg1", "tmp_"]
```

Negation is defined to bring about an `"e"` closure and so ordinarily an indefinite will scope under negation.

(24)
```
(S (NP-SBJ (PRP I))
   (VP (VBP do) (RB n't)
       (VP (VB like) (NP (NNS enchiladas)))) (. .))
```

(25)
```
( λlc.
  ( ( ( pro lc fh "i") "arg0")
    ( neg
      ( ( ( some lc fh "e" ( nn lc "enchiladas")) "arg1")
        ( verb lc ["arg1"] "like"))))) ["h", "arg0", "arg1"]
```

(26) $\neg\exists e_1 x(\text{enchiladas}(x) \wedge \text{like}(e_1, \text{Speaker}, x))$

Definites are distinguishable from indefinites on the grounds that their binding value always comes from a `"d"` closure and so a definite will always scope with the discourse level (and so have a treatment essentially like the indefinite of (23)). Moreover any indefinite within the restriction of a definite is implemented to scope with the definite at the discourse level, as (27)–(29) illustrate.

(27)
```
(S (NP-SBJ (PRP I))
   (VP (VBP do) (RB n't)
       (VP (VB like)
           (NP (NP (DT the) (NN taste))
               (PP (IN of) (NP (NNS enchiladas)))))) (. .))
```

(28)
```
( λlc.
  ( ( ( pro lc fh "i") "arg0")
    ( neg
      ( ( ( dt lc fh "the" "d"
            ( ( ( prep "of_")
                ( some lc fh "d" ( nn lc "enchiladas")))
              ( nn lc "taste"))) "arg1")
        ( verb lc ["arg1"] "like")))))
["h", "arg0", "arg1", "of_"]
```

(29) $\exists xy(\text{enchiladas}(x) \wedge$
$\text{is\_taste\_of}(y, x) \wedge \neg\exists e_1 \text{like}(e_1, \text{Speaker}, y))$

Discourse level scoping of an indefinite as a consequence of falling within the restriction of a definite is however prevented if some other scope taking element intervenes within the restriction of the definite,

as seen with (30) where negation falls between "the young" and the instances of "cows" and "hand", so that "cows" and "hand" scope with negation in (32).

(30)
```
(S (NP-SBJ (NP (DT The) (JJ young))
           (SBAR (WHNP-2 (WP who))
                 (S (NP-SBJ (-NONE- *T*-2))
                    (VP (MD wo) (RB n't)
                        (VP (VB milk)
                            (NP (NNS cows))
                            (PP-MNR (IN by) (NP (NN hand))))))))
   (VP (VBP leave)))
```

(31)
```
( λlc. ( ( ( dt lc fh "the" "d"
           ( ( coord fh " ∧ "
               ( relc
                 ( ( ( wpro) "h")
                   ( clause
                     ( ( ( trace) "arg0")
                       ( neg
                         ( ( ( some lc fh "e" ( nn lc "cows"))
                             "arg1")
                           ( ( ( prep "mnr_by_")
                               ( some lc fh "e" ( nn lc "hand")))
                             ( verb lc ["arg1", "mnr_by_"]
                                                  "milk")))))))))
         ( nn lc "young"))) "arg0")
  ( present ( verb lc nil "leave")))) ["h", "arg0", "arg1", "mnr_by_"]
```

(32) $\exists x(\text{young}(x) \wedge$
$\neg\exists e_2 yz(\text{cows}(z) \wedge \text{hand}(y) \wedge \text{milk}(e_2, x, z) \wedge$
$\text{mnr\_by}(e_2) = y) \wedge \exists e_1(e_1 \approx e_0 \wedge \text{leave}(e_1, x)))$

6 Results

Having a system of evaluation as the basis for generating meaning representations makes possible accepting input with minimal alteration to syntactic information that is known to be correct and on a scale equal to the availability of treebanks. To explore the coverage this allows the system has been used to automatically convert into corresponding banks of meaning representations content from The Penn Treebank-3 (LDC99T42; Mitchell P. Marcus and Taylor 1999), The Brown-GENIA Treebank (Lease and Charniak 2005), and The PennBioIE 0.9 Treebank (Kulick et al. 2004), with the coverage results of table 1.

Treebank	Corpus	no. of trees	fully completed	partially completed	failed
Penn Treebank-3	ATIS-3	580	580 (100%)	0	0
	SWBD	110505	108216 (97.93%)	1072 (0.97%)	1217 (1.1%)
	WSJ	49208	48764 (99.1%)	412 (0.84%)	32 (0.06%)
Brown-GENIA	-	210	205 (97.6%)	5 (2.4%)	0
PennBioIE 0.9	cyp	3392	3347 (98.67%)	19 (0.56%)	26 (0.77%)
	onco	3030	3002 (99.08%)	17 (0.56%)	11 (0.36%)

Table 1: Coverage results

To give a sense of how the meaning representations have scaled to the data we illustrate three examples.

Example (33) from the WSJ corpus of the Penn Treebank demonstrates coverage of an embedded clause. Interesting aspects of the representation to note include the definite description "the end of the year" that has widest scope despite occurring with an indefinite ("exports") that scopes inside the embedded clause. Also, the tense of the embedded clause is made relative to the tense of the embedding clause.

(33) Government officials said exports at the end of the year would remain under a government target of $68 billion.

(34)
$$\begin{array}{l}\exists xy(\text{year}(x) \wedge \text{is\_end\_of}(y,x) \wedge \\ \quad \exists e_1(e_1 < e_0 \wedge \\ \quad\quad \exists z(\text{government\_officials}(z) \wedge \\ \quad\quad\quad \text{said}(e_1,z, \\ \quad\quad\quad\quad \exists e_2 uv(\text{exports}(v) \wedge \text{time}(v) \sqsubseteq \text{at}(y) \wedge \\ \quad\quad\quad\quad\quad \$68\_\text{BILLION}x_1(\text{unit}(x_1), \\ \quad\quad\quad\quad\quad\quad \text{is\_government\_target\_of}(u,x_1)) \wedge \\ \quad\quad\quad\quad\quad \exists e_3(e_2 \approx e_1 \wedge e_2 < e_3 \wedge \\ \quad\quad\quad\quad\quad\quad \text{remain}(e_3,v) \wedge \text{location}(e_3) \sqsubseteq \text{under}(u)))))))\end{array}$$

From the Brown-GENIA corpus (35) includes the representation of an embedded clause and the modal "may" as an operator that scopes over a proposition.

(35) Increased expression of wild-type NFAT1 substantially increases IL-4 promoter activity in unprimed CD4 T cells, suggesting NFAT1 may be limiting for IL-4 gene expression in this cell type.

(36)
$$\begin{array}{l}\exists x(\text{cell\_type}(x) \wedge \\ \quad \exists e_1 e_2(e_2 \approx e_0 \wedge \\ \quad\quad \exists yzuv(\text{wild-type}(y) \wedge \text{NFAT1}(y) \wedge \\ \quad\quad\quad \text{increased}(v) \wedge \text{is\_expression\_of}(v,y) \wedge \\ \quad\quad\quad\quad \text{IL-4\_promoter\_activity}(u) \wedge \\ \quad\quad\quad\quad\quad \text{unprimed}(z) \wedge \text{CD4-T\_cells}(z) \wedge \\ \quad\quad\quad\quad\quad \text{suggesting}(e_1,v, \\ \quad\quad\quad\quad\quad\quad \exists x_1(\text{NFAT1}(x_1) \wedge \\ \quad\quad\quad\quad\quad\quad\quad \text{may}(\exists e_3 x_2(\text{is\_IL-4\_gene\_expression\_in}(x_2,x) \wedge \\ \quad\quad\quad\quad\quad\quad\quad\quad\quad \text{limiting}(e_3,x_1) \wedge \text{for}(e_3) = x_2)))) \wedge \\ \quad\quad\quad\quad\quad\quad \text{increases}(e_2,v,u) \wedge \text{location}(e_2) \sqsubseteq \text{in}(z) \wedge \\ \quad\quad\quad\quad\quad\quad \exists e_3(e_3 = e_2 \wedge \text{substantially}(e_3)))))\end{array}$$

Example (38) from the WSJ corpus of the Penn Treebank demonstrates coverage of discourse. In the coding of the passives ("fired" and "prosecuted") in (39) "_" fills the agent argument slot of the predicate to mark that there is no agent binding.

(38) Mrs. Yeargin was fired and prosecuted under an unusual South Carolina law that makes it a crime to breach test security. In September, she pleaded guilty and paid a $500 fine. Her alternative was 90 days in jail. Her story is partly one of personal downfall. She was an unstinting teacher who won laurels and inspired students, but she will probably never teach again.

(39)
$$
\begin{aligned}
&\exists p_1 e_1 e_2 e_3 e_4 e_5 e_6 e_7 (e_2 < e_0 \wedge e_3 < e_0 \wedge e_4 < e_0 \wedge \\
&e_5 < e_0 \wedge e_6 \approx e_0 \wedge e_7 < e_0 \wedge \\
&\exists x y z u v x_1 x_2 x_3 x_4 x_5 x_6 (\exists e_8 (e_8 \approx e_0 \wedge \\
&\quad \text{unusual}(x) \wedge \text{South\_Carolina\_law}(x) \wedge \\
&\quad \text{makes}(e_8, x, \\
&\quad\quad \exists e_9 e_{10} x_7 x_8 x_9 x_{10} (\text{crime}(x_{10}) \wedge \\
&\quad\quad\quad \exists x_{11} (\text{choose3}(x_{11}, x, \text{Mrs. Yeargin}, e_1) \wedge \\
&\quad\quad\quad\quad \text{crime}(x_8) \wedge \\
&\quad\quad\quad\quad\quad \exists x_{12} (\text{choose5}(x_{12}, x_8, x_{11}, x, \text{Mrs. Yeargin}, e_1) \wedge \\
&\quad\quad\quad\quad\quad\quad \text{test\_security}(x_7) \wedge \text{breach}(e_9, x_{12}, x_7))) \wedge \\
&\quad\quad\quad\quad \exists x_{11} (\exists x_{12} (\text{choose8}(x_{12}, x_{10}, x_{11}, e_9, x_8, x_7, x, \text{Mrs. Yeargin}, \\
&\quad\quad\quad\quad\quad e_1) \wedge \text{test\_security}(x_9) \wedge \text{breach}(e_{10}, x_{12}, x_9)))))) \wedge \\
&\exists x_7 (\text{choose8}(x_7, \text{September}, \text{Mrs. Yeargin}, e_4, e_3, e_2, e_1, y, x) \wedge \\
&\quad \text{is\_alternative\_of}(v, x_7)) \wedge \text{jail}(z) \wedge \\
&\quad 90(u) \wedge \text{days}(u) \wedge \text{location}(u) \sqsubseteq \text{in}(z) \wedge \\
&\quad \exists x_7 (\text{choose12}(x_7, \text{September}, \text{Mrs. Yeargin}, e_5, e_4, e_3, e_2, e_1, v, \\
&\quad\quad u, z, y, x) \wedge \text{is\_story\_of}(x_3, x_7)) \wedge \\
&\quad\quad \text{personal}(x_1) \wedge \text{downfall}(x_1) \wedge \\
&\quad\quad\quad \text{is\_one\_of}(x_2, x_1) \wedge \\
&\quad\quad\quad\quad \text{fired}(e_1, \_, \text{Mrs. Yeargin}) \wedge \\
&\quad\quad\quad\quad\quad \text{prosecuted}(e_2, \_, \text{Mrs. Yeargin}) \wedge \text{under}(e_2) = x \wedge \\
&\quad\quad\quad\quad\quad \exists x_7 (\text{choose4}(x_7, \text{Mrs. Yeargin}, e_2, e_1, x) \wedge \\
&\quad\quad\quad\quad\quad\quad \text{fine}(y) \wedge 500(y) \wedge \text{unit}(y) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \exists x_8 (\text{guilty}(x_8) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad \text{pleaded}(e_3, x_7, x_8) \wedge \text{time}(e_3) \sqsubseteq \text{in}(\text{September})) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad \text{paid}(e_4, x_7, y) \wedge \text{time}(e_4) \sqsubseteq \text{in}(\text{September})) \wedge \\
&\quad\quad\quad\quad\quad\quad \text{was}(e_5, v, u) \wedge \text{is}(e_6, x_3, x_2) \wedge \text{partly}(e_6) \wedge \\
&\quad\quad\quad\quad\quad\quad \exists x_7 (\text{choose16}(x_7, \text{September}, \text{Mrs. Yeargin}, e_6, e_5, e_4, e_3, \\
&\quad\quad\quad\quad\quad\quad\quad e_2, e_1, x_3, x_2, x_1, v, u, z, y, x) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \exists e_8 e_9 (e_8 < e_0 \wedge e_9 < e_0 \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad \text{unstinting}(x_6) \wedge \text{teacher}(x_6) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{laurels}(x_4) \wedge \text{students}(x_5) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{won}(e_8, x_6, x_4) \wedge \text{inspired}(e_9, x_6, x_5)) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \text{was}(e_7, x_7, x_6)) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \exists x_7 (\text{choose20}(x_7, e_7, x_6, x_5, x_4, \text{September}, \text{Mrs. Yeargin}, \\
&\quad\quad\quad\quad\quad\quad\quad e_6, e_5, e_4, e_3, e_2, e_1, x_3, x_2, x_1, v, u, z, y, x) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \text{will}(\text{probably}(\text{never}(\exists e_8 (\text{teach}(e_8, x_7) \wedge \text{again}(e_8)))))))))
\end{aligned}
$$

The automatic encoding of pronouns found in (39) introduces an existentially bound variable and an $n+1$-place "choose$n$" predicate, as in (40), such that to resolve the pronoun the existentially bound x might be equated to one of the variables y_1,...,y_n that form the range of accessible antecedents calculated at the stage of evaluation.

(40) $\exists x(\text{choose}n(x,y_1,...,y_n) \wedge ...)$

To give an idea of more general properties of the banks of semantic representations we have generated we can, for example, measure the frequency of aspects of the representations.

Meaning representations generated with the WSJ data contain 49004 distinct predicates with event arguments, 15131 of these appear more than once, 4234 appear more than twice, and 1939 appear five times or more. The frequency distribution for predicates with event arguments is given in table 2.

predicate frequency			number of predicates
1000	$\leq f <$	6200	6
100	$\leq f <$	1000	89
10	$\leq f <$	100	1839
5	$\leq f <$	10	2295
2	$\leq f <$	5	10897
0	$< f \leq$	1	33878

Table 2: Frequency distribution of predicates with event arguments using WSJ data

Table 3 details the 20 predicates with the highest frequency together with arguments.

frequency	predicate and arguments (event argument suppressed)
6114	said (arg0 entity) (that proposition)
2593	is (arg1 entity) (arg0 entity)
2034	says (arg0 entity) (that proposition)
1395	are (arg1 entity) (arg0 entity)
1353	was (arg1 entity) (arg0 entity)
1088	be (arg1 entity) (arg0 entity)
779	have (arg1 entity) (arg0 entity)
643	say (arg0 entity) (that proposition)
590	has (arg1 entity) (arg0 entity)
581	were (arg1 entity) (arg0 entity)
499	is (arg0 entity) (that proposition)
414	had (arg1 entity) (arg0 entity)
392	expected (arg0 entity) (that proposition)
376	include (arg1 entity) (arg0 entity)
325	have (arg0 entity) (that proposition)
299	think (arg0 entity) (that proposition)
259	make (arg1 entity) (arg0 entity)
257	said (arg0 entity) (location (in entity)) (that proposition)
253	expects (arg0 entity) (that proposition)
252	want (arg0 entity) (that proposition)

Table 3: Top 20 highest frequency predicates with event arguments using WSJ data

Tables 4 and 5 detail results for predicates with event arguments in the meaning representations generated using the PennBioIE data.

predicate frequency			number of predicates
100	$\leq f <$	250	8
10	$\leq f <$	100	226
5	$\leq f <$	10	337
2	$\leq f <$	5	1616
0	$< f \leq$	1	3374

Table 4: Frequency distribution of predicates with event arguments using PennBioIE data

frequency	predicate and arguments (event argument suppressed)
231	using (arg1 entity) (arg0 entity)
230	was (arg1 entity) (arg0 entity)
215	is (arg1 entity) (arg0 entity)
190	are (arg1 entity) (arg0 entity)
166	were (arg1 entity) (arg0 entity)
151	had (arg1 entity) (arg0 entity)
138	suggest (arg0 entity) (that proposition)
123	inhibited (arg1 entity) (arg0 entity)
96	involved (in entity) (arg1 entity)
92	inhibited (arg0 entity) (arg1 entity)
88	showed (arg1 entity) (arg0 entity)
87	associated (with entity) (arg1 entity)
75	found (location (in entity)) (arg1 entity)
74	indicate (arg0 entity) (that proposition)
68	be (arg1 entity) (arg0 entity)
67	inhibit (arg1 entity) (arg0 entity)
65	have (arg1 entity) (arg0 entity)
53	found (that proposition)
51	detected (location (in entity)) (arg1 entity)
49	suggesting (arg0 entity) (that proposition)

Table 5: Top 20 highest frequency predicates with event arguments using PennBioIE data

We can also use the generated meaning representations to count differing scope structures. With the meaning representations generated from the WSJ data we identified 7234 scope structures. Of these 3860 appear more than once, and 2244 appear five times or more. The frequency distribution for scope structures is shown in table 6.

structure frequency			number of structures
10000	$\leq f <$	18000	2
1000	$\leq f <$	10000	1
100	$\leq f <$	1000	23
10	$\leq f <$	100	190
5	$\leq f <$	10	201
2	$\leq f <$	5	871
0	$< f \leq$	1	5946

Table 6: Frequency distribution of scope structures with WSJ data

Table 7 presents the top 20 highest frequency scope structures with the WSJ data, with '<' used to indicate 'scopes over'. By far the most frequent is an essentially flat scope structure with the meaning representation being build from only existential quantification and conjunction. The WSJ data does however contain a considerable number of embedding taking predicates, e.g., *said*, *expected*, *think*, seen already with table 3 and grouped together in table 7 as THAT. Possible cases of scope ambiguity with quantifiers arise considerably lower on the frequency scale, and indeed we have to develop a more crafted extraction of scope structure to explore this with any reliability.

frequency	scope structure
17717	
10489	THAT
1140	¬
648	THAT < ¬
619	∀
508	∨
434	THAT < ?
399	ABOUT
287	?
285	¬ < THAT
243	THAT < ∀
237	THAT < ¬
229	THAT < ABOUT
198	THAT < ∨
193	can
192	MORE_THAN
185	may
175	∀ < THAT
173	could
124	? < THAT

Table 7: Top 20 highest frequency scope structures with WSJ data

Results for scope structures using the PennBioIE data are presented in tables 8 and 9.

structure frequency			number of structures
1000	$\leq f <$	3900	1
100	$\leq f <$	1000	3
10	$\leq f <$	100	24
5	$\leq f <$	10	21
2	$\leq f <$	5	81
0	$< f \leq$	1	535

Table 8: Frequency distribution of scope structures with PennBioIE data

frequency	quantificational structure
3816	
581	THAT
264	¬
198	∨
99	∀
73	may
56	THAT < may
44	¬ ∨
44	THAT < ?
34	THAT < ¬
33	can
28	EACH
27	could
22	ABOUT
21	¬ < ANY
21	THAT < can
20	?
15	THAT < ∨
14	¬ < ¬
14	∨ < ¬

Table 9: Top 20 highest frequency scope structures with PennBioIE data

We can also provide statistics for the SCT expressions used as the basis for the evaluations that generate meaning representations. In this regard a notable property is the presence of grammatical binding names which productively arise with the conversion of functional tag and preposition information.

With the SCT expressions converted from the WSJ corpus there is the creation of 728 binding names. Of these 319 only appear once, and 534 appear less than five times. Figure 1 illustrates the growth of binding names with the WSJ data. The numbers seem to have converged for names that appear at least twice or more while new binding names that appear only once keep appearing. Some binding names that appear only once are due to noise from the treebank annotation or conversion process, but the overwhelming majority are linguistically significant, e.g., *tmp_starting_with*, *prp_primarily_because_of*, *loc_well_below*, *dir_out_from_under*, *instead_of_through*, *following_on*, *conditioned_on*, *based_loc_on*, *as_compared_with*, *accounting_for*.

With the SCT expressions converted from the PennBioIE corpus there is the creation of 225 binding names. Of these 73 only appear once, and 153 appear less than five times. Figure 2 illustrates the name growth, with an essentially identical pattern emerging to figure 1.

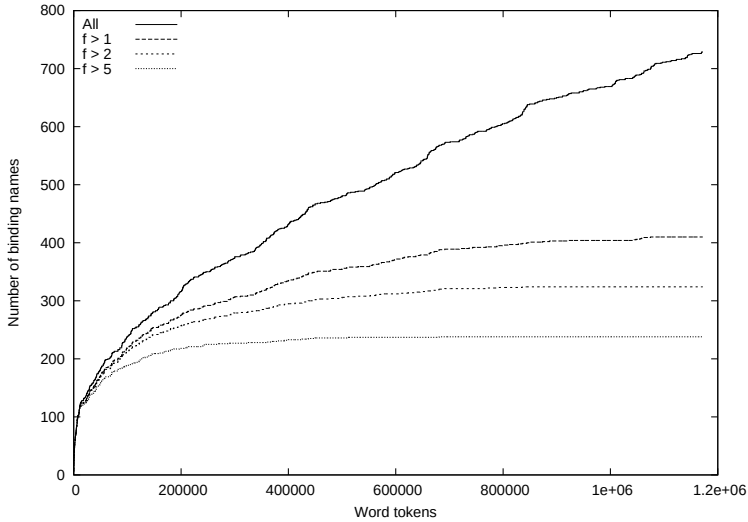


Figure 1: Growth of binding names using WSJ data

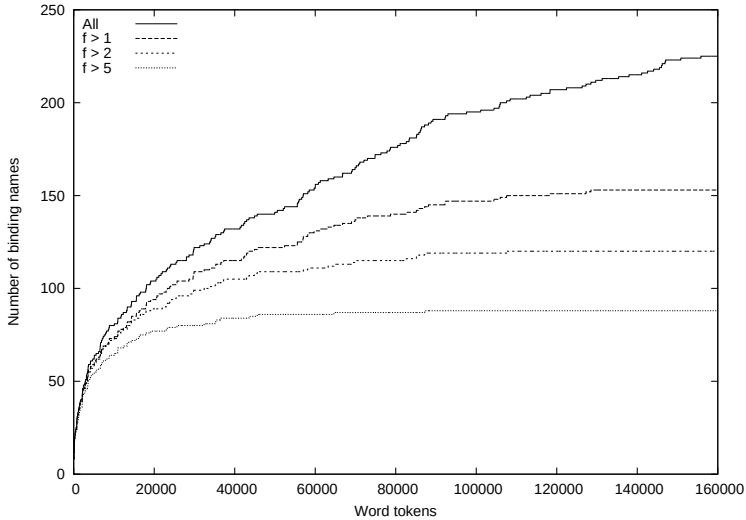


Figure 2: Growth of binding names using PennBioIE data

7 Related Work

As first steps for getting to meaning representations there are methods for taking the Penn Treebank and extracting Lexicalized Tree-Adjoining Grammars, (Xia et al. 2000) Combinatory Categorial Grammars (CCG) (Hockenmaier 2003) and Head-driven Phrase Structure

Grammars (HPSG) (Miyao and Tsujii 2008). Extracted grammars have led to the development of parsers that allow assembling meaning representations during parse time: Curran et al. (2007) and Basile and Bos (2011) employ a parser derived from an extracted CCG grammar to build semantic representations in the framework of Discourse Representation Theory, while Sato et al. (2006) build semantic representations of Typed Dynamic Logic with an HPSG based parser.

Considering partial semantic representations, Cahill et al. (2002) present an algorithm to annotate the Penn Treebank with Lexical Functional Grammar F-structures, and Spreyer and Frank (2005) apply to the TIGER Dependency Bank for German a method for obtaining banks of underspecified meaning representations from dependency structures. Also of relevance is the Redwoods approach to treebanking (Oepen et al. 2002) in which analyses are recorded as defined by the LinGO English Resource Grammar, which includes generating underspecified meaning representations.

However the current authors are unaware of gold standard fully specified meaning representations for any sizeable amount of data, and worse still of established methods by which meaning representations might be adequately compared and evaluated, although Bos (2008) contains suggestions. It is hoped that the current work in allowing the building of large banks of meaning representations can contribute towards the establishment of some benchmarks in this area.

8 Summary

To sum up we have demonstrated a method for building banks of meaning representations from treebanks. Since the system is able to work directly with gold standard treebanks it is able to inherit the advantage of a syntactic base that is known to be essentially correct, an unavoidable requirement for building meaning representations of any suitability. The system has allowed the construction of a number of banks of meaning representations from existing treebanks with both high coverage and quality of representation in preserving and making fully explicit dependencies through generating scoped operations of quantification and their bindings.

Acknowledgments

This research has been supported by the JST PRESTO program (Synthesis of Knowledge for Information Oriented Society).

References

Basile, Valerio and Johan Bos. 2011. Towards generating text from discourse representation structures. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, pages 145–150. Nancy, France.

Bies, Ann, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Tech. Rep. MS-CIS-95-06, LINC LAB 281, University of Pennsylvania Computer and Information Science Department.

Bos, Johan. 2008. Let's not argue about semantics. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008)*. Marrakech, Morocco.

Butler, Alastair. 2010. *The Semantics of Grammatical Dependencies*, vol. 23 of *Current Research in the Semantics/Pragmatics Interface*. Bingley: Emerald.

Cahill, Aoife, Mairéad McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation—Bootstrapping Annotated Language Data*, Las Palmas, Spain, pages 8–15.

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 Demonstrations Session (ACL-07 demo)*, pages 33–36.

Davidson, Donald. 1967. The logical form of action sentences. In N. Rescher, ed., *The Logic of Decision and Action*. Pittsburgh: University of Pittsburgh Press. Reprinted in: D. Davidson, 1980. *Essays on Actions and Events*. Claredon Press, Oxford, pages 105–122.

Hockenmaier, Julia. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Kulick, Seth, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, and Lyle Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of HLT/NAACL 2004*, pages 61–68. Association for Computational Linguistics.

Landman, Fred. 2000. *Events and Plurality: The Jerusalem Lectures*. Dordrecht: Kluwer Academic Publishers.

Lease, Matthew and Eugene Charniak. 2005. Parsing biomedical literature. In *Second International Joint Conference on Natural Language Processing (IJCNLP'05)*. Springer-Verlag.

Marcus, Michell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, Beatrice Santorini and Ann Taylor. 1999. Treebank-3. Linguistic Data Consortium, Philadelphia.

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

Oepen, Stephan, E. Callahan, Dan Flickinger, and Christoper D. Manning. 2002. LinGO Redwoods. A rich and dynamic treebank for HPSG. In *LREC workshop on parser evaluation*. Las Palmas, Spain.

Santorini, Beatrice. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). Tech. Rep. Tech Report MS-CIS-90-47, Linc Lab 178, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Sato, Manabu, Daisuke Bekki, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Translating HPSG-style outputs of a robust parser into Typed Dynamic Logic. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. Sydney, Australia.

Spreyer, Kathrin and Anette Frank. 2005. The TIGER 700 RMRS Bank: RMRS Construction from Dependencies. In *Proceedings of the 6th International Workshop on Linguistically Interpreted Corpora*, pages 1–10. Jeju Island, Korea.

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

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *In Proceedings of the 2000 Conference on Empirical Methods in Natural Language Processing*, pages 53–62. Hong Kong.