Translating HPSG-style Outputs of a Robust Parser into Typed Dynamic Logic

Manabu Sato[†] **Daisuke Bekki**[‡] **Yusuke Miyao**[†] **Jun'ichi Tsujii**[†∗]
† Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan
‡ Center for Evolutionary Cognitive Sciences, University of Tokyo
Komaba 3-8-1, Meguro-ku, Tokyo 153-8902, Japan
∗School of Informatics, University of Manchester
PO Box 88, Sackville St, Manchester M60 1QD, UK
∗SORST, JST (Japan Science and Technology Corporation)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012, Japan
† {sa-ma, yusuke, tsujii}@is.s.u-tokyo.ac.jp
‡ bekki@ecs.c.u-tokyo.ac.jp

Abstract

The present paper proposes a method by which to translate outputs of a robust HPSG parser into semantic representations of Typed Dynamic Logic (TDL), a dynamic plural semantics defined in typed lambda calculus. With its higher-order representations of contexts, TDL analyzes and describes the inherently inter-sentential nature of quantification and anaphora in a strictly lexicalized and compositional manner. The present study shows that the proposed translation method successfully combines robustness and descriptive adequacy of contemporary semantics. The present implementation achieves high coverage, approximately 90%, for the real text of the Penn Treebank corpus.

1 Introduction

Robust parsing technology is one result of the recent fusion between symbolic and statistical approaches in natural language processing and has been applied to tasks such as information extraction, information retrieval and machine translation (Hockenmaier and Steedman, 2002; Miyao et al., 2005). However, reflecting the field boundary and unestablished interfaces between syntax and semantics in formal theory of grammar, this fusion has achieved less in semantics than in syntax.

For example, a system that translates the output of a robust CCG parser into semantic representations has been developed (Bos et al., 2004). While its corpus-oriented parser attained high coverage with respect to real text, the expressive power of the resulting semantic representations is confined to first-order predicate logic.

The more elaborate tasks tied to discourse information and plurality, such as resolution of anaphora antecedent, scope ambiguity, presupposition, topic and focus, are required to refer to 'deeper' semantic structures, such as dynamic semantics (Groenendijk and Stokhof, 1991).

However, most dynamic semantic theories are not equipped with large-scale syntax that covers more than a small fragment of target languages. One of a few exceptions is Minimal Recursion Semantics (MRS) (Copestake et al., 1999), which is compatible with large-scale HPSG syntax (Pollard and Sag, 1994) and has affinities with UDRS (Reyle, 1993). For real text, however, its implementation, as in the case of the ERG parser (Copestake and Flickinger, 2000), restricts its target to the static fragment of MRS and yet has a lower coverage than corpus-oriented parsers (Baldwin, to appear).

The lack of transparency between syntax and discourse semantics appears to have created a tension between the robustness of syntax and the descriptive adequacy of semantics.

In the present paper, we will introduce a robust method to obtain dynamic semantic representations based on Typed Dynamic Logic (TDL) (Bekki, 2000) from real text by translating the outputs of a robust HPSG parser (Miyao et al., 2005). Typed Dynamic Logic is a dynamic plural semantics that formalizes the structure underlying the semantic interactions between quantification, plurality, bound variable/E-type anaphora

$$
\begin{array}{lcl}
r^{e\times\cdots\times e\mapsto t}x_1^i\cdots x_n^i & \equiv & \lambda G^{(i\mapsto e)\mapsto t}.\lambda g^{i\mapsto e}.g\in G\wedge r\langle gx_1,\ldots,gx_m\rangle \\
\sim\phi^{prop} & \equiv & \lambda G^{(i\mapsto e)\mapsto t}.\lambda g^{i\mapsto e}.g\in G\wedge\neg\exists h^{i\mapsto e}.h\in\phi G \\
\begin{bmatrix}\phi^{prop}\\ \vdots \\ \varphi^{prop}\end{bmatrix} & \equiv & \lambda G^{(i\mapsto e)\mapsto t}.(\varphi\cdots(\phi G)) \\
ref\left(x^i\right)[\phi^{prop}][\varphi^{prop}] & \equiv & \lambda G^{(i\mapsto e)\mapsto t}.\left\{\begin{array}{l} if\ \ {}^{G}\!/_{x}={}^{\phi G}\!/_{x} \\ then\ \lambda g^{i\mapsto e}.g\in\varphi G\wedge\ {}^{G}\!/_{x}={}^{\varphi G}\!/_{x} \\ otherwise\ undefined\end{array}\right\}
\end{array}
$$

$$
\left(\begin{array}{llcl}
where & prop & \equiv & ((i\mapsto e)\mapsto t)\mapsto(i\mapsto e)\mapsto t \\
 & g^{\alpha}\in G^{\alpha\mapsto t} & \equiv & Gg \\
 & G^{(i\mapsto e)\mapsto t}\Big/_{x^i} & \equiv & \lambda d^{e}.\exists g^{i\mapsto e}.g\in G\wedge gx=d
\end{array}\right)
$$

Figure 1: Propositions of TDL (Bekki, 2005)

and presuppositions. All of this complex discourse/plurality-related information is encapsulated within higher-order structures in TDL, and the analysis remains strictly lexical and compositional, which makes its interface with syntax transparent and straightforward. This is a significant advantage for achieving robustness in natural language processing.

2 Background

2.1 Typed Dynamic Logic

Figure 1 shows a number of propositions defined in (Bekki, 2005), including atomic predicate, negation, conjunction, and anaphoric expression. Typed Dynamic Logic is described in typed lambda calculus (Gödel's System T) with four ground types: e(entity), i(index), n(natural number), and t(truth). While assignment functions in static logic are functions in meta-language from type e variables (in the case of first-order logic) to objects in the domain D_e, assignment functions in TDL are functions in object-language from indices to entities. Typed Dynamic Logic defines the notion *context* as a set of assignment functions (an object of type $(i\mapsto e)\mapsto t$) and a *proposition* as a function from context to context (an object of type $((i\mapsto e)\mapsto t)\mapsto(i\mapsto e)\mapsto t$). The conjunctions of two propositions are then defined as composite functions thereof. This setting conforms to the view of "propositions as information flow", which is widely accepted in dynamic semantics.

Since all of these higher-order notions are described in lambda terms, the path for compositional type-theoretic semantics based on functional application, functional composition and type raising is clarified. The derivations of TDL semantic representations for the sentences "A boy ran. He tumbled." are exemplified in Figure 2 and Figure 3. With some instantiation of variables, the semantic representations of these two sentences are simply conjoined and yield a single representation, as shown in (1).

$$
(1)\quad \begin{bmatrix} boy'x_1s_1 \\ run'e_1s_1 \\ agent'e_1x_1 \\ ref(x_2)[\,]\begin{bmatrix} tumble'e_2s_2 \\ agent'e_2x_2\end{bmatrix}\end{bmatrix}
$$

The propositions $boy'x_1s_1$, $run'e_1s_1$ and $agent'e_1x_1$ roughly mean "the entity referred to by x_1 is a boy in the situation s_1", "the event referred to by e_1 is a running event in the situation s_1", and "the agent of event e_1 is x_1", respectively.

The former part of (1) that corresponds to the first sentence, filtering and testing the input context, returns the updated context schematized in (2). The updated context is then passed to the latter part, which corresponds to the second sentence as its input.

(2)

$\cdots$	x_1	s_1	e_1	$\cdots$
	john	$situation_1$	$running_1$	
	john	$situation_2$	$running_2$	
	$\vdots$	$\vdots$	$\vdots$	

This mechanism makes anaphoric expressions, such as "He" in "He tumbles", accessible to its preceding context; namely, the descriptions of their presuppositions can refer to the preceding context compositionally. Moreover, the referents of the anaphoric expressions are correctly calculated as a result of previous filtering and testing.

$$\dfrac{\dfrac{\dfrac{\text{“a”}}{\begin{array}{l}\lambda n^{i\mapsto i\mapsto p\mapsto p}.\lambda w^{i\mapsto i\mapsto i\mapsto p\mapsto p}.\\ \lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.nx_1 s\left[wx_1 es\phi\right]\end{array}} \quad \dfrac{\text{“boy”}}{\lambda x^{i}.\lambda s^{i}.\lambda \phi^{p}.\begin{bmatrix} boy'xs \\ \phi \end{bmatrix}}}{\lambda w^{i\mapsto i\mapsto i\mapsto p\mapsto p}.\lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.\begin{bmatrix} boy'x_1 s \\ wx_1 es\phi \end{bmatrix}} \quad \dfrac{\text{“ran”}}{\begin{array}{l}\lambda sbj^{(i\mapsto i\mapsto i\mapsto p\mapsto p)\mapsto i\mapsto i\mapsto p\mapsto p}.\\ sbj\left(\lambda x^{i}.\lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.\begin{bmatrix} run'es \\ agent'ex \\ \phi \end{bmatrix}\right)\end{array}}}{\lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.\begin{bmatrix} boy'x_1 s_1 \\ run'es \\ agent'ex_1 \\ \phi \end{bmatrix}}$$

Figure 2: Derivation of a TDL semantic representation of “A boy ran”.

$$\dfrac{\dfrac{\text{“he”}}{\begin{array}{l}\lambda w^{i\mapsto i\mapsto i\mapsto p\mapsto p}.\\ \lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.ref\left(x_2\right)[\,]\left[wx_2 es\phi\right]\end{array}} \quad \dfrac{\text{“tumbled”}}{\begin{array}{l}\lambda sbj^{(i\mapsto i\mapsto i\mapsto p\mapsto p)\mapsto i\mapsto i\mapsto p\mapsto p}.\\ sbj\left(\lambda x^{i}.\lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.\begin{bmatrix} tumble'es \\ agent'ex \\ \phi \end{bmatrix}\right)\end{array}}}{\lambda e^{i}.\lambda s^{i}.\lambda \phi^{p}.ref\left(x_2\right)[\,]\begin{bmatrix} tumble'e_2 s_2 \\ agent'e_2 x_2 \end{bmatrix}}$$

Figure 3: Derivation of TDL semantic representation of “He tumbled”.

Although the antecedent for x_2 is not determined in this structure, the possible candidates can be enumerated: x_1, s_1 and e_1, which precede x_2. Since TDL seamlessly represents linguistic notions such as “entity”, “event” and “situation”, by indices, the anaphoric expressions, such as “the event” and “that case”, can be treated in the same manner.

2.2 Head-driven Phrase Structure Grammar

Head-driven Phrase Structure Grammar (Pollard and Sag, 1994) is a kind of lexicalized grammar that consists of lexical items and a small number of composition rules called schema. Schemata and lexical items are all described in typed feature structures and the unification operation defined thereon.

$$(3)\quad \begin{bmatrix} PHON & \text{“boy”} \\ \begin{array}{l} SYN \\ SEM \end{array} & \begin{bmatrix} HEAD & \begin{bmatrix} noun & \\ MOD & \langle\ \rangle \end{bmatrix} \\ VAL & \begin{bmatrix} SUBJ & \langle\ \rangle \\ COMPS & \langle\ \rangle \\ SPR & \langle det \rangle \end{bmatrix} \\ SLASH & \langle\ \rangle \end{bmatrix} \end{bmatrix}$$

Figure 4 is an example of a parse tree, where the feature structures marked with the same boxed numbers have a shared structure. In the first stage of the derivation of this tree, lexical items are assigned to each of the strings, “John” and “runs.” Next, the mother node, which dominates the two items,

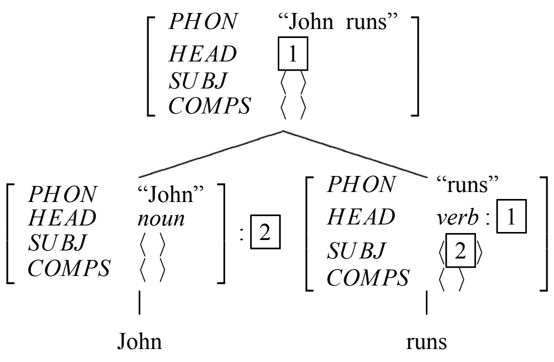

Figure 4: An HPSG parse tree

is generated by the application of Subject-Head Schema. The recursive application of these operations derives the entire tree.

3 Method

In this section, we present a method to derive TDL semantic representations from HPSG parse trees, adopting, in part, a previous method (Bos et al., 2004). Basically, we first assign TDL representations to lexical items that are terminal nodes of a parse tree, and then compose the TDL representation for the entire tree according to the tree structure (Figure 5). One problematic aspect of this approach is that the composition process of TDL semantic representations and that of HPSG parse trees are not identical. For example, in the HPSG

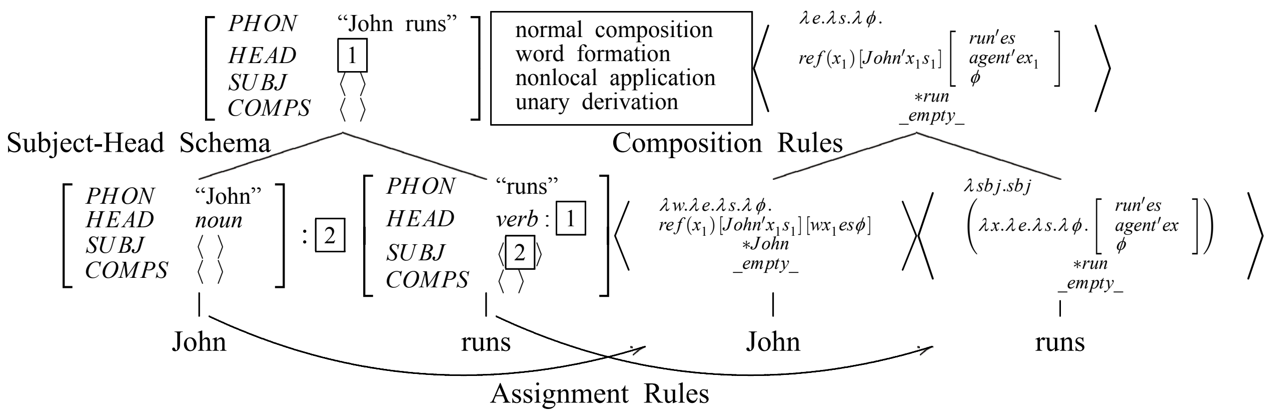

Figure 5: Example of the application of the rules

parser, a compound noun is regarded as two distinct words, whereas in TDL, a compound noun is regarded as one word. Long-distance dependency is also treated differently in the two systems. Furthermore, TDL has an operation called unary derivation to deal with empty categories, whereas the HPSG parser does not have such an operation.

In order to overcome these differences and realize a straightforward composition of TDL representations according to the HPSG parse tree, we defined two extended composition rules, **word formation rule** and **non-local application rule**, and redefined TDL unary derivation rules for the use in the HPSG parser. At each step of the composition, one composition rule is chosen from the set of rules, based on the information of the schemata applied to the HPSG tree and TDL representations of the constituents. In addition, we defined extended TDL semantic representations, referred to as **TDL Extended Structures (TDLESs)**, to be paired with the extended composition rules.

In summary, the proposed method is comprised of TDLESs, assignment rules, composition rules, and unary derivation rules, as will be elucidated in subsequent sections.

3.1 Data Structure

A TDLES is a tuple $\langle T,p,n\rangle$, where T is an extended TDL term, which can be either a TDL term or a special value ω. Here, ω is a value used by the **word formation rule**, which indicates that the word is a *word modifier* (See Section 3.3). In addition, p and n are the necessary information for extended composition rules, where p is a matrix predicate in T and is used by the **word formation rule**, and n is a nonlocal argument, which takes either a variable occurring in T or an empty value. This element corresponds to the SLASH feature in HPSG and is used by the **nonlocal application rule**.

The TDLES of the common noun "boy" is given in (4). The contents of the structure are T, p and n, beginning at the top. In (4), T corresponds to the TDL term of "boy" in Figure 2, p is the predicate *boy*, which is identical to a predicate in the TDL term (the identity relation between the two is indicated by "$*$"). If either T or p is changed, the other will be changed accordingly. This mechanism is a part of the **word formation rule**, which offers advantages in creating a new predicate from multiple words. Finally, n is an empty value.

(4) $$\left\langle \begin{array}{c} \lambda x.\lambda s.\lambda\phi.\begin{bmatrix} *boy'xs \\ \phi \end{bmatrix} \\ *boy \\ \_empty\_ \end{array} \right\rangle$$

3.2 Assignment Rules

We define assignment rules to associate HPSG lexical items with corresponding TDLESs. For closed class words, such as "a", "the" or "not", assignment rules are given in the form of a template for each word as exemplified below.

(5) $$\begin{array}{c} \begin{bmatrix} PHON & \text{"a"} \\ HEAD & det \\ SPEC & \langle noun \rangle \end{bmatrix} \\ \Downarrow \\ \left\langle \begin{array}{c} \lambda x.\lambda s.\lambda\phi.\begin{bmatrix} \lambda n.\lambda w.\lambda e.\lambda s.\lambda\phi. \\ nx_1 s\,[wx_1 es\phi] \end{bmatrix} \\ \_empty\_ \\ \_empty\_ \end{array} \right\rangle \end{array}$$

Shown in (5) is an assignment rule for the indefinite determiner “a”. The upper half of (5) shows a template of an HPSG lexical item that specifies its phonetic form as “a”, where POS is a determiner and specifies a noun. A TDLES is shown in the lower half of the figure. The TDL term slot of this structure is identical to that of “a” in Figure 2, while slots for the matrix predicate and nonlocal argument are empty.

For open class words, such as nouns, verbs, adjectives, adverbs and others, assignment rules are defined for each syntactic category.

(6) $$\begin{bmatrix} PHON & P \\ HEAD & noun \\ MOD & \langle\rangle \\ SUBJ & \langle\rangle \\ COMPS & \langle\rangle \\ SPR & \langle det \rangle \end{bmatrix}$$
$$\Downarrow$$
$$\left\langle \begin{array}{c} \lambda x.\lambda s.\lambda \phi.\begin{bmatrix} *P'xs \\ \phi \end{bmatrix} \\ *P \\ \_empty\_ \end{array} \right\rangle$$

The assignment rule (6) is for common nouns. The HPSG lexical item in the upper half of (6) specifies that the phonetic form of this item is a variable, *P*, that takes no arguments, does not modify other words and takes a specifier. Here, POS is a noun. In the TDLES assigned to this item, an actual input word will be substituted for the variable *P*, from which the matrix predicate *P*′ is produced. Note that we can obtain the TDLES (4) by applying the rule of (6) to the HPSG lexical item of (3).

As for verbs, a base TDL semantic representation is first assigned to a verb root, and the representation is then modified by lexical rules to reflect an inflected form of the verb. This process corresponds to HPSG lexical rules for verbs. Details are not presented herein due to space limitations.

3.3 Composition Rules

We define three composition rules: **the function application rule**, **the word formation rule**, and **the nonlocal application rule**. Hereinafter, let $S_L = \langle T_L, p_L, n_L \rangle$ and $S_R = \langle T_R, p_R, n_R \rangle$ be TDLESs of the left and the right daughter nodes, respectively. In addition, let S_M be TDLESs of the mother node.

Function application rule: The composition of TDL terms in the TDLESs is performed by function application, in the same manner as in the original TDL, as explained in Section 2.1.

Definition 3.1 (function application rule). *If* $Type(T_L) = \alpha$ *and* $Type(T_R) = \alpha \mapsto \beta$ *then*

$$S_M = \left\langle \begin{array}{c} T_R T_L \\ p_R \\ union(n_L, n_R) \end{array} \right\rangle$$

Else if $Type(T_L) = \alpha \mapsto \beta$ *and* $Type(T_R) = \alpha$ *then*

$$S_M = \left\langle \begin{array}{c} T_L T_R \\ p_L \\ union(n_L, n_R) \end{array} \right\rangle$$

In Definition 3.1, $Type(T)$ is a function that returns the type of TDL term T, and $union(n_L, n_R)$ is defined as:

$$union(n_L, n_R) = \begin{cases} empty & if\ n_L = n_R = \_empty\_ \\ n & if\ n_L = n,\ n_R = \_empty\_ \\ n & if\ n_L = \_empty\_,\ n_R = n \\ undefined & if\ n_L \neq \_empty\_,\ n_R \neq \_empty\_ \end{cases}$$

This function corresponds to the behavior of the union of SLASH in HPSG. The composition in the right-hand side of Figure 5 is an example of the application of this rule.

Word formation rule: In natural language, it is often the case that a new word is created by combining multiple words, for example, “orange juice”. This phenomenon is called *word formation*. Typed Dynamic Logic and the HPSG parser handle this phenomenon in different ways. Typed Dynamic Logic does not have any rule for word formation and regards “orange juice” as a single word, whereas most parsers treat “orange juice” as the separate words “orange” and “juice”. This requires a special composition rule for word formation to be defined. Among the constituent words of a compound word, we consider those that are not HPSG heads as *word modifiers* and define their value for T as ω. In addition, we apply the **word formation rule** defined below.

Definition 3.2 (word formation rule). *If* $Type(T_L) = \omega$ *then*

$$S_M = \left\langle \begin{array}{c} T_R \\ concat(p_L, p_R) \\ n_R \end{array} \right\rangle$$

Else if $Type(T_R) = \omega$ *then*

$$S_M = \left\langle \begin{array}{c} T_L \\ concat(p_L, p_R) \\ n_L \end{array} \right\rangle$$

$concat(p_L, p_R)$ in Definition 3.2 is a function that returns a concatenation of p_L and p_R. For example, the composition of a word modifier "orange" (7) and and a common noun "juice" (8) will generate the TDLES (9).

(7) $$\left\langle \begin{array}{c} \omega \\ orange \\ \_empty\_ \end{array} \right\rangle$$

(8) $$\left\langle \begin{array}{c} \lambda x.\lambda s.\lambda \phi. \begin{bmatrix} *juice'xs \\ \phi \end{bmatrix} \\ *juice \\ \_empty\_ \end{array} \right\rangle$$

(9) $$\left\langle \begin{array}{c} \lambda x.\lambda s.\lambda \phi. \begin{bmatrix} *orange\_juice'xs \\ \phi \end{bmatrix} \\ *orange\_juice \\ \_empty\_ \end{array} \right\rangle$$

Nonlocal application rule: Typed Dynamic Logic and HPSG also handle the phenomenon of wh-movement differently. In HPSG, a wh-phrase is treated as a value of SLASH, and the value is kept until the *Filler-Head Schema* are applied. In TDL, however, wh-movement is handled by the functional composition rule.

In order to resolve the difference between these two approaches, we define the **nonlocal application rule**, a special rule that introduces a slot relating to HPSG SLASH to TDLESs. This slot becomes the third element of TDLESs. This rule is applied when the *Filler-Head Schema* are applied in HPSG parse trees.

Definition 3.3 (nonlocal application rule). *If* $Type(T_L) = (\alpha \mapsto \beta) \mapsto \gamma$, $Type(T_R) = \beta$, $Type(n_R) = \alpha$ *and the Filler-Head Schema are applied in HPSG, then*

$$S_M = \left\langle \begin{array}{c} T_L(\lambda n_R.T_R) \\ p_L \\ \_empty\_ \end{array} \right\rangle$$

3.4 Unary Derivation Rules

In TDL, type-shifting of a word or a phrase is performed by composition with an empty category (a category that has no phonetic form, but has syntactic/semantic functions). For example, the phrase "this year" is a noun phrase at the first stage and can be changed into a verb modifier when combined with an empty category. Since many of the type-shifting rules are not available in HPSG, we defined unary derivation rules in order to provide an equivalent function to the type-shifting rules of TDL. These unary rules are applied independently with HPSG parse trees. (10) and (11) illustrate the unary derivation of "this year". (11) is derived from (10) using a unary derivation rule.

Table 1: Number of implemented rules

assignment rules	
HPSG-TDL template	51
for closed words	16
for open words	35
verb lexical rules	27

composition rules	
binary composition rules	3
function application rule	
word formation rule	
nonlocal application rule	
unary derivation rules	12

(10)
$$\left\langle \begin{array}{c} \lambda w.\lambda e.\lambda s.\lambda \phi.ref(x_1)[*year'x_1s_1][wx_1es\phi] \\ *year \\ \_empty\_ \end{array} \right\rangle$$

(11)
$$\left\langle \begin{array}{c} \lambda v.\lambda e.\lambda s.\lambda \phi. \\ ref(x_1)[*year'x_1s_1]\left[ves\begin{bmatrix} mod'ex_1 \\ \phi \end{bmatrix}\right] \\ *year \\ \_empty\_ \end{array} \right\rangle$$

4 Experiment

The number of rules we have implemented is shown in Table 1. We used the Penn Treebank (Marcus, 1994) Section 22 (1,527 sentences) to develop and evaluate the proposed method and Section 23 (2,144 sentences) as the final test set.

We measured the coverage of the construction of TDL semantic representations, in the manner described in a previous study (Bos et al., 2004). Although the best method for strictly evaluating the proposed method is to measure the agreement between the obtained semantic representations and the intuitions of the speaker/writer of the texts, this type of evaluation could not be performed because of insufficient resources. Instead, we measured the rate of successful derivations as an indicator of the coverage of the proposed system.

The sentences in the test set were parsed by a robust HPSG parser (Miyao et al., 2005), and HPSG parse trees were successfully generated for 2,122 (98.9%) sentences. The proposed method was then applied to these parse trees. Table 2 shows that 88.3% of the un-

Table 2: Coverage with respect to the test set

covered sentences	88.3 %
uncovered sentences	11.7 %
assignment failures	6.2 %
composition failures	5.5 %
word coverage	99.6 %

Table 3: Error analysis: the development set

# assignment failures	103
# unimplemented words	61
# TDL unsupporting words	17
# nonlinguistic HPSG lexical items	25
# composition failures	72
# unsupported compositions	20
# invalid assignments	36
# nonlinguistic parse trees	16

seen sentences are assigned TDL semantic representations. Although this number is slightly less than 92.3%, as reported by Bos et al., (2004), it seems reasonable to say that the proposed method attained a relatively high coverage, given the expressive power of TDL.

The construction of TDL semantic representations failed for 11.7% of the sentences. We classified the causes of the failure into two types. One of which is application failure of the assignment rules (assignment failure); that is, no assignment rules are applied to a number of HPSG lexical items, and so no TDLESs are assigned to these items. The other is application failure of the composition rules (composition failure). In this case, a type mismatch occurred in the composition, and so a TDLES was not derived.

Table 3 shows further classification of the causes categorized into the two classes. We manually investigated all of the failures in the development set.

Assignment failures are caused by three factors. Most assignment failures occurred due to the limitation in the number of the assignment rules (as indicated by "unimplemented words" in the table). In this experiment, we did not implement rules for infrequent HPSG lexical items. We believe that this type of failure will be resolved by increasing the number of assignment rules. The second factor in the table, "TDL unsupported words", refers to expressions that are not covered by the current theory of TDL. In order to resolve this type of failure, the development of TDL is required. The third factor, "nonlinguistic HPSG lexical items" includes a small number of cases in which TDLESs are not assigned to the words that are categorized as nonlinguistic syntactic categories by the HPSG parser. This problem is caused by ill-formed outputs of the parser.

```
ref($1)[]
   [lecture($2,$3) &
    past($3) &
    agent($2,$1) &
    content($2,$4) &
    ref($5)[]
       [every($6)[ball($6,$4)]
              [see($7,$4) &
               present($4) &
               agent($7,$5) &
               theme($7,$6) &
               tremendously($7,$4) &
               ref($8)[]
                   [ref($9)[groove($9,$10)]
                        [be($11,$4) &
                         present($4) &
                         agent($11,$8) &
                         in($11,$9) &
                         when($11,$7)]]]]]
```

Figure 6: Output for the sentence: *"When you're in the groove, you see every ball tremendously," he lectured.*

The composition failures can be further classified into three classes according to their causative factors. The first factor is the existence of HPSG schemata for which we have not yet implemented composition rules. These failures will be fixed by extending of the definition of our composition rules. The second factor is type mismatches due to the unintended assignments of TDLESs to lexical items. We need to further elaborate the assignment rules in order to deal with this problem. The third factor is parse trees that are linguistically invalid.

The error analysis given above indicates that we can further increase the coverage through the improvement of the assignment/composition rules.

Figure 6 shows an example of the output for a sentence in the development set. The variables `$1, ...,$11` are indices that represent entities, events and situations. For example, `$3` represents a situation and `$2` represents the lecturing event that exists in `$3`. `past($3)` requires that the situation is past. `agent($2,$1)` requires that the entity `$1` is the agent of `$2`. `content($2,$4)` requires that `$4` (as a set of possible worlds) is the content of `$2`. `be($11,$4)` refers to `$4`. Finally, `every($6)[ball($6,$4)][see($7,$4) ...]` represents a generalized quantifier "every ball". The index `$6` serves as an antecedent both for bound-variable anaphora within its scope and for E-type anaphora outside its scope. The entities that correspond to the two occurrences of "you" are represented by `$8` and `$5`. Their unification is left as an anaphora resolution task that can be easily solved by existing statistical or rule-based methods, given the structural information of the TDL semantic representation.

5 Conclusion

The present paper proposed a method by which to translate HPSG-style outputs of a robust parser (Miyao et al., 2005) into dynamic semantic representations of TDL (Bekki, 2000). We showed that our implementation achieved high coverage, approximately 90%, for real text of the Penn Treebank corpus and that the resulting representations have sufficient expressive power of contemporary semantic theory involving quantification, plurality, inter/intra-sentential anaphora and presupposition.

In the present study, we investigated the possibility of achieving robustness and descriptive adequacy of semantics. Although previously thought to have a trade-off relationship, the present study proved that robustness and descriptive adequacy of semantics are not intrinsically incompatible, given the transparency between syntax and discourse semantics.

If the notion of robustness serves as a criterion not only for the practical usefulness of natural language processing but also for the validity of linguistic theories, then the compositional transparency that penetrates all levels of syntax, sentential semantics, and discourse semantics, beyond the superficial difference between the laws that govern each of the levels, might be reconsidered as an essential principle of linguistic theories.

References

Timothy Baldwin, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim and Stephan Oepen (to appear) Beauty and the Beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar ? and the corpus, In *Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives*, Mouton de Gruyter.

Daisuke Bekki. 2000. Typed Dynamic Logic for Compositional Grammar, Doctoral Dissertation, University of Tokyo.

Daisuke Bekki. 2005. Typed Dynamic Logic and Grammar: the Introduction, manuscript, University of Tokyo,

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran and Julia Hockenmaier. 2004. Wide-Coverage Semantic Representations from a CCG Parser, In *Proc. COLING '04*, Geneva.

Ann Copestake, Dan Flickinger, Ivan A. Sag and Carl Pollard. 1999. Minimal Recursion Semantics: An introduction, manuscript.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG In *Proc. LREC-2000*, Athens.

Jeroen Groenendijk and Martin Stokhof. 1991. Dynamic Predicate Logic, In *Linguistics and Philosophy 14*, pp.39-100.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring Compact Lexicalized Grammars from a Cleaner Treebank, In *Proc. LREC-2002*, Las Palmas.

Mitch Marcus. 1994. The Penn Treebank: A revised corpus design for extracting predicate-argument structure. In *Proceedings of the ARPA Human Language Technolog Workshop*, Princeton, NJ.

Yusuke Miyao, Takashi Ninomiya and Jun'ichi Tsujii. 2005. Corpus-oriented Grammar Development for Acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank, in *IJCNLP 2004, LNAI3248*, pp.684-693. Springer-Verlag.

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

Uwe Reyle. 1993. Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction, In *Journal of Semantics 10*, pp.123-179.