Linguistic Phenomena, Analyses, and Representations: Understanding Conversion between Treebanks

Rajesh Bhatt
Univ. of Massachusetts
Amherst, MA 01003, USA
bhatt@linguist.umass.edu

Owen Rambow
CCLS, Columbia University
New York, NY 10115, USA
rambow@ccls.columbia.edu

Fei Xia
Univ. of Washington
Seattle, WA 98195, USA
fxia@uw.edu

Abstract

Treebanks are valuable resources for natural language processing (NLP). There is much work in NLP which converts treebanks from one representation (e.g., phrase structure) to another (e.g., dependency) before applying machine learning. This paper provides a framework in which to think about the question of when such a conversion is possible.

1 Introduction

There has been much interest in converting treebanks from one representation to another; for instance, from phrase structure to dependency structure (e.g., motivated by the recent surge in interest in dependency parsing), or from phrase structure to other grammatical frameworks such as LTAG, HPSG, CCG, or LFG. While there has been much work on converting between treebank representations (Collins et al., 1999; Xia and Palmer, 2001; Cahill et al., 2002; Nivre, 2003; Hockenmaier and Steedman, 2007), there has not been a general yet precise discussion of what conditions are necessary for such conversion to happen.

In this paper, we provide an analytical framework for determining how difficult it would be to convert representations under one set of annotation guidelines M_1 to representations under another set of guidelines M_2. We are only interested in cases where annotation guidelines are available for both levels of representation, since it is not clear how one would interpret an undocumented representation, and thus it would not be clear how to evaluate the conversion results. Given two sets of guidelines and a particular linguistic phenomenon, there are three possible scenarios: (1) the phenomenon is represented only on one side; (2) the phenomenon is represented on both sides with incompatible analyses; (3) the phenomenon is represented on both sides with compatible analyses. We give a formal definition of *compatibility* and a procedure for distinguishing these three scenarios. We also discuss how each scenario will affect automatic conversion. Using this framework, researchers can determine the difficulty of a conversion task between existing guidelines, or they can design guidelines for new treebanks so automatic conversion to other representations can be as smooth as possible.

Note that we are not addressing general questions such as "In general, is it easier to convert dependency to phrase structure or *vice versa*?" We believe that such general questions cannot be answered. One needs to examine what information is being represented before the issue of conversion can be addressed, i.e., we must first study the guidelines of the two levels of representation.

While we propose a general approach to analyzing syntactic representations, throughout the paper we will use examples based on converting dependency structures to phrase structures. Specifically, we will use as a source of examples the Hindi/Urdu Treebank (HUTB) (Palmer et al., 2009). The HUTB is unusual in that it contains a dependency structure (DS) annotation, a PropBank-style annotation (PB) (Kingsbury et al., 2002) for predicate-argument structure, and an independently motivated phrase-structure (PS) annotation which is automatically derived from DS plus PB. For lack of space, we will not discuss the PropBank layer in this paper and instead draw all examples from PS and DS.

The structure of the paper is as follows. Section 2 introduces some terminology which helps in our analysis. Section 3 discusses the notion of compatibility and syntactic consistency. Section 4 introduces a procedure for comparing two sets of annotation guidelines with respect to conversion. Section 5 discuss examples from the HUTB that fall into the two "harder" scenarios for conversion.

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 1234–1242,
Chiang Mai, Thailand, November 8 – 13, 2011. ©2011 AFNLP

2 Important concepts in a treebank

This study focuses on the relation between DS and PS treebanks. To understand whether an automatic conversion between DS and PS is possible, it is important to distinguish a few concepts in a treebank. Following (Rambow, 2010), we distinguish three concepts: the linguistic phenomena (what he calls "content"), the representation type, and the linguistic theory (what he calls "syntactic theory"). We reinterpret these concepts and extend them, in terms of the HUTB.

2.1 Linguistic phenomena

The **linguistic phenomena** are what we want to represent about the words which make up our treebank: they are the reason for treebanking. If there were no interesting linguistic phenomena, there would be no reason to create treebanks. The task of treebanking consists of identifying which of the phenomena of interest appear in a given sequence of words (a data token) and then to choose the correct representation for these phenomena in the given data token. The types of linguistic phenomena range from general concepts such as recursive constituency (which words in this sentence form phrases?) to types of relations between words or between a word and a phrase (e.g., subjecthood, or temporal modification) to specific constructions (e.g., small clauses). Linguistic phenomena also include finer-grained distinctions within coarser categories (e.g., unergative/unaccusative as two classes of intransitive verbs). For all of these phenomena, while linguists may disagree about the proper representation or whether the phenomenon is present in a particular instance, they typically agree on the fact that the phenomenon exists in the language, or exists in some language.

Consider first the example of syntactic constituency. There is broad agreement among syntacticians that syntax groups words recursively into hierarchies; to our knowledge, no serious syntactic theory uses only flat representations (such as base phrases). Crucially, this is independent of whether the syntactician uses DS or PS: DS also assumes a recursive structure and represents constituency (in a DS, each subtree represents a constituent, headed by its root).[1] It is difficult to assume that a treebank (DS or PS) would not represent syntactic constituency – there would have to be an explicit disclaimer that what looks like constituents (in DS or PS) are in fact not linguistically meaningful units, and are just notationally expedient devices.

Now consider the example of an embedded small clauses, as in the English sentence *Atif considered Seema stupid*, or its Hindi counterpart in (1). This is a particular construction; it is characterized (in both English and Hindi) by the fact that the NP *Seema* is an argument of the predicate *stupid*, but its case and word order is that of an object of the main verb *considered*, not a subject (as can be seen if we replace it with a pronoun, *her*).

(1)	Atif-ne	Seema-ko	bewakuuf	samjhaa
	Atif-Erg	Seema-Acc	stupid	consider.Pfv

'Atif considered Seema stupid.'

2.2 Representation type

The **representation type** is the type of mathematical object that is used to represent syntactic facts. A DS is a tree in which all nodes are labeled with words or empty strings (e.g., empty categories). A PS is a tree in which all and only the leaf nodes are labeled with words or empty strings, and the internal nodes are labeled with nonterminal symbols (e.g., syntactic labels). In addition, each representation type can decide what more specific representation devices it will employ, such as labels on the arcs of a tree (e.g., dependency type in a DS), or the use of empty nodes, or coindexation between nodes (e.g., to mark syntactic movement).

2.3 Linguistic theory

A **formal linguistic description** explains how linguistic phenomena are represented in the chosen representation type; a formal description is thus tied to a particular representation type. It can be thought of as a mapping from linguistic phenomena to linguistic representations in the chosen representation type. It has two components: a theoretical framework, and linguistic analyses. If, in addition, the analyses provided by a formal description are such that they rule out certain strings in the language and make falsifiable predictions, then we call the formal linguistic description a **linguistic theory**. These notions of "formal linguistic description" and of "linguistic theory" should not be confused with a **theoretical framework**, such as

[1]Of course, PS allows for intermediate projections. These have two functions. First, they distinguish functionally distinct dependents, such as subject from object. Second, an intermediate projection may actually occur as an empirically identifiable constituent, as in VP fronting in English. In both cases, DS can use alternate representational devices. We leave a fuller discussion to future work.

Government and Binding (GB) or LFG. The goal of theoretical framework is not to provide a complete description of a single language, but rather to provide vocabulary and constraints in which linguistic theories can be formulated.

Once a linguistic theory has chosen a theoretical framework such as GB, the next step is to determine how to represent the linguistic phenomena in that framework. For instance, given the Hindi embedded small clause example in (1), there are many possible ways to represent the phenomenon in a PS-based linguistic theory (e.g., the ones in Figure (1a-c)) or in a DS-based linguistic theory (e.g., the ones in Figure (1d-f)). We call them different **analyses** of this phenomenon.

It is important to stress that elements of the representation on their own may have no meaning. For example, the trace **CASE** in (1c) is not meaningful in isolation. Instead, the trace and its coindexed partner, the NP *Seema ko*, along with their structural configuration, together signify the phenomenon (which we are calling embedded small clause) that was identified by the annotator as happening in this particular sentence. The annotator chose this way of representing the phenomenon for this data token because the annotation guidelines say to do so. But the annotation of course also manifests the particular analysis chosen (namely, the raising-to-object analysis of (1c) and (1f)). However, this analysis is *not* specific to this particular data token; rather, for all annotations that use the guidelines, it must be used whenever an embedded small clause is identified by the annotator. The annotator cannot identify an embedded small clause but suddenly change the analysis on his or her own. It is also impossible that annotation guidelines would identify a unified phenomenon and propose two analyses based on arbitrary conditions (say, the first letter of the head noun). Thus, annotators must learn how to represent each phenomenon, and then must decide which phenomena a specific data token exhibits.

2.4 Annotation guidelines

Every treebank requires annotation guidelines, which can be regarded as a formal linguistic description, typically a very detailed and explicit one with descriptions and examples. The guidelines are used to train annotators, for annotators as a reference, and for users of the treebank as a guide to its meaning. Some annotation guidelines may even be linguistic theories (if they can be used to make predictions about ungrammatical sentences in the language, for example), though this is not generally the case.

To create annotation guidelines, the guideline designers need to choose a theoretical framework and a set of linguistic phenomena to be captured. Next, they need to determine a linguistic analysis for each linguistic phenomenon, and demonstrate the analysis with descriptions and examples (e.g., sentences and the corresponding DS or PS trees).

Take the HUTB as an example. Because it contains both representation types, DS and PS, it has two sets of guidelines for syntactic annotation, one for each representation type. The DS annotation guidelines follow the Paninian grammatical model (Bharati et al., 1995; Begum et al., 2008). The PS guidelines are inspired by the Principles-and-Parameters methodology, as instantiated by the theoretical developments starting with Government and Binding Theory (Chomsky, 1981).

3 Compatibility and conversion

As mentioned in the previous section, annotation guidelines provide linguistic analyses for a set of linguistic phenomena, and they are tied to a representation type (DS or PS). Now given two sets of annotation guidelines (one for DS and the other for PS), the central question is whether automatic conversion between DS and PS is possible; that is, is it possible to write a conversion algorithm that takes as input a DS tree annotated according to the DS guidelines, and produces a PS tree that would be correct according to the PS guidelines, or vice versa? In the rest of the paper, we will focus on the DS-to-PS conversion.

The answer to the question depends on the guidelines. If the DS and PS guidelines cover the same set of linguistic phenomena (explicitly or implicitly) and they choose *compatible* analyses for the phenomena, automatic conversion is possible. If these conditions do not hold, automatic conversion would require additional information or mechanism, as explained in Section 5. In this section, we will provide a formal definition of *compatibility*.

3.1 Intuition about compatibility

To define compatibility between linguistic analyses, let us first look at an example. Figure 1 shows several analyses for small clause: three for PS and

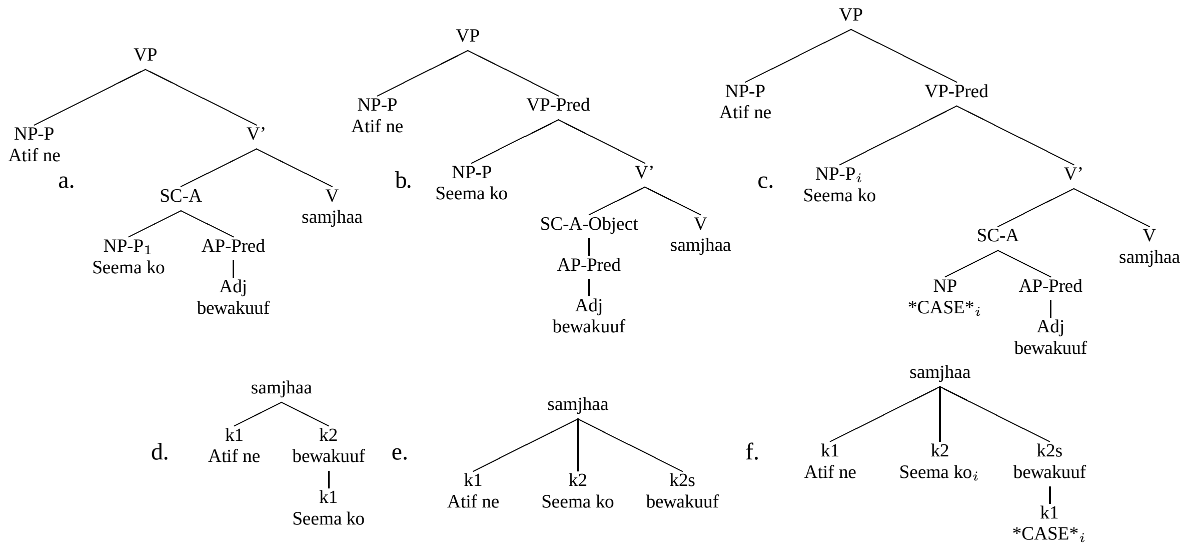

Figure 1: Possible analyses for the Hindi small clause example in Ex (1)

three for DS. It is clear that the analyses in (a) and (d) have something in common (the "exceptional case-marking" analysis), in which the semantic relationship between the adjectival predicated *bewakuuf ('stupid')* and *Seema ko* is seen as primary and the source of the object case marking *ko* on *Seema ko* is not represented explicitly. Similarly, (b) and (e) share an analysis, in which the presence of the object case marking *ko* is seen as primary, and predicate-argument relation between *Seema ko* and *bewakuuf ('stupid')* is deduced only from the label *SC-A-Object* in (b) or *k2s* in (e). Finally, the trees in (c) and (f) share an analysis (the "raising-to-object" analysis), in which a trace is used to indicate that the NP *Seema ko* participates in two relations.

Intuitively, analyses in (a) and (d) are compatible, so are the ones in (b) and (e), and the ones in (c) and (f). The next question is whether we can provide a formal definition of compatibility and write code that automatically checks whether the DS and PS analyses for a linguistic phenomenon is compatible. The answer is affirmative, as we can do that via the definition of *consistency* between (DS, PS) tree pairs, as is explained below.

3.2 Implicit vs. explicit information

Before we define *consistency*, there are two points that are worth mentioning. First, DS and PS, as two representation types, use different representation devices to describe syntactic structure: DS uses edges to represent the dependency or modifier-modifiee relation between words, whereas PS uses internal nodes to mark the spans and types of syntactic constituents. As a result, there are certain aspects of information that DS has to provide *explicitly* but PS does not need to (e.g., DS has to mark the direction of each edge, indicating which node is the head and which node is the dependent). The converse is also true (e.g., each internal node in a PS has to be labeled, indicating the syntactic category of the phrase).

Second, not explicitly providing certain information does not mean that the corresponding concept does not exist in the syntactic theory. For instance, PS does not need to mark the head of an internal node explicitly, but it does not mean that the syntactic theory chosen for PS does not have the concept of *headedness*.

3.3 Syntactic consistency

Our definition of consistency assumes that each phrase in a PS has a special word called *head word* which represents the main properties of the phrase, an assumption shared by all major contemporary syntactic frameworks. A pair (DS, PS) of DS and PS trees for the same sentence is called *consistent* if there is a way to assign a head word to each internal node in the PS so that all the words in the subtree rooted at that internal node are descendants of the head word in the DS. A formal definition is given later, but let us start with an example.

Figure 2 shows a simple PS with two internal nodes and three leaf nodes. Because the head words for the internal nodes are not marked in the PS, there are several possibilities in choosing the

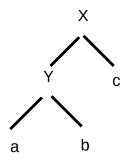

Figure 2: A simple PS: *a*, *b*, and *c* are leaf nodes, *X* and *Y* are internal nodes

head words for the internal nodes: the head word of the Y can be *a* or *b*, and the head word of X can be *c* or the head word of Y, resulting in four possible DSs, as shown in Figure 3. In contrast, no matter which head words we choose for the internal nodes in the PS, the resulting DSs will not be the ones in Figure 4. We call the DSs in Figure 3 *consistent* with the PS, and the DSs in Figure 4 *inconsistent* with the PS.

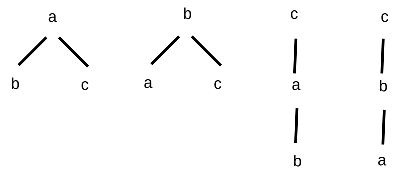

Figure 3: The DSs *consistent* with PS in Fig. 2

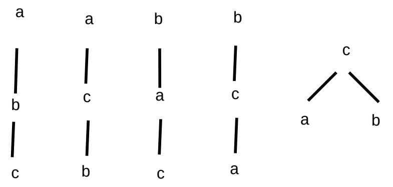

Figure 4: The DSs *inconsistent* with PS in Fig. 2

More formally, let us define two operations on a PS. Given a PS and an assignment of head words for the internal nodes in the PS, a *flatten* operation recursively merges each internal node X with its head child (a head child is a node which has the same head word as its parent). When two nodes, *X* and its head child *h*, are merged, the other children of *X* and the children of *h* (if any) become the children of the new merged node. Then, a *label replacement* operation replaces the label of each internal node with the node's head word. For instance, given the PS in Figure 2 and the assignment where a is the head word of Y and c is the head word of X, the tree after the flatten operation is in Figure 5(ii), and the tree after the label replacement operation is in Figure 5(iii). A PS and a DS are called *consistent* if and only if there exists an assignment of head words for the internal nodes in PS such that after the flatten operation and the label replacement operation, the new PS is identical to the DS.

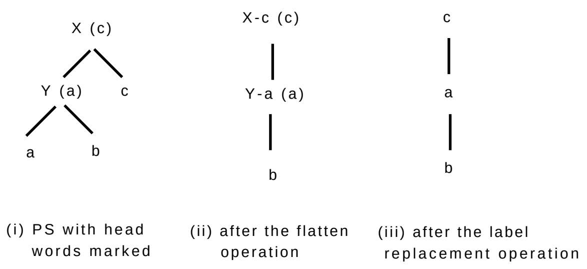
X (c)
Y (a)
c
a
b
X-c (c)
Y-a (a)
b
c
a
b
(i) PS with head words marked
(ii) after the flatten operation
(iii) after the label replacement operation

Figure 5: The resulting PS after the flatten and label replacement operations: *X(c)* in (i) means that *c* is the head word of *X*; *X-c* in (ii) means the nodes *X* and *c* are merged.

Given a (DS, PS) pair, one can use the following process to check whether the DS and the PS are consistent. For each edge, ($head$, dep), in the DS, find the nodes for $head$ and dep in the PS and their closest common ancestor $ancest$; for each node on the path between $head$ and $ancest$ (including $ancest$), assign $head$ as its head word; for each node on the path between dep and $ancest$ (excluding $ancest$), assign dep as its head word. The DS and PS are consistent *iff* after all the edges in the DS have been used, each internal node in the PS is assigned exactly one head word.

Now we can define the notion of *compatible analyses*. Given a linguistic phenomenon, let D be the set of (DS, PS) pairs provided in the guidelines for that phenomenon. The analyses in the DS and PS guidelines are *compatible* if and only if every (DS, PS) pair in D is *consistent*.

3.4 Conversion between DS and PS

Given a DS, there are multiple PSs that are consistent with the DS. The reason that a DS-to-PS conversion algorithm could make the right selection is that the (DS,PS) pairs in the annotation guidelines indicate what a PS should look like for a given DS. For instance, Figure 6 shows some patterns in the (DS,PS) pairs: the first pattern says that when a noun depends on a verb with the type *SBJ* in a DS, the corresponding PS should include a *S* node which has two children, an *NP* node that dominates the noun and a *VP* node that dominates the verb. The meaning of the second pattern can be interpreted similarly. Xia et al. (2009) showed that such patterns can be learned from (DS, PS) pairs automatically and with these patterns their conversion algorithm produced good results when tested on the English Penn Treebank.

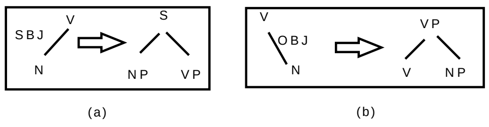

Figure 6: Two patterns that could help a DS-to-PS conversion to produce the correct PS tree as output

4 Analytic Framework for Comparing Treebank Guidelines

We now present our procedure for comparing two sets of treebank guidelines, with the goal of determining whether automatic conversion is possible. The devil is in the details. It is impossible to read the introduction to two sets of annotation guidelines and then to be able to say whether automatic conversion is possible. Instead, it is necessary to look at every single phenomenon: one phenomenon may be easy to convert, while another may be quite hard. We illustrate our procedure assuming we want to transform DS into PS.

For each linguistic phenomenon Φ, we ask two questions: (1) is Φ captured in both DS and PS guidelines? (2) if so, are the analyses in DS and PS guidelines compatible? The answers to the questions lead to three scenarios:

- **The phenomenon is represented by both sides and the analyses are compatible**: automatic conversion can be done using the procedure presented in Section 3, using knowledge which is general to Φ.
- **The phenomenon is represented by both sides but the analyses are incompatible**: automatic conversion is possible but it requires additional mechanisms (e.g., *DS+* as introduced in Section 5.1.2) to bridge the gap in analyses. The knowledge needed is general to Φ.
- **The phenomena is represented only on one side**: If it is represented in the DS only, the conversion algorithm can simply ignore it when creating PS. If it is represented on the PS side only, automatic conversion will require additional information which is not general to Φ, but which provides information specific to each **instance** of Φ (for example, a list of unaccusative verbs, as used in Section 5.2.1).

Of course, establishing the range of phenomena to be considered may not be entirely trivial. It includes not only the set of all constructions in a narrow sense, but also which constituents are represented, which empty arguments are included, what types of dependencies are represented, and so on. We discuss some examples in the following section.

5 Preliminary results in HUTB

As a case study, we compared the PS and DS guidelines of the HUTB with the process outlined in Section 4. The guidelines currently include 209 sentences where both DS and PS trees are provided. Each sentence has a sentence id, which indicates the linguistic phenomenon the sentence intends to represent. We ran the consistency check algorithm on the (DS, PS) pairs and found that 162 out of 209 pairs are consistent.

We then used the consistency results to group the corresponding phenomena into one of the three categories in Section 4. It turns out that most phenomena belong to the first category. For the other two categories, we present one example below and discuss how that will affect conversion.

5.1 Phenomena represented on both sides but differently

This category comprises several constructions in the HUTB: long scrambling and extraposition (which are non-projective), small clauses, local scrambling, and support verb constructions. We discuss small clauses in detail as a typical case.

5.1.1 Small clause

In HUTB, both the DS and the PS analysis represent the sharing aspect of small clauses, but they do so differently, which leads to incompatibility. In the PS analysis, as in Figure (1c), *Seema* is interpreted as the argument of the predicate *bewakuuf* ('stupid') and hence, given the theoretical assumptions adopted by the PS guidelines, it must combine with this predicate. But it gets case from the matrix predicate and hence also has a relationship with the matrix predicate. As a result, *Seema-ko* corresponds to two positions in the PS tree: a lower position (the empty category

CASE) as the subject of the lower predicate and a higher position as the object of the higher predicate. The two positions and the coindexation between them indicate the movement of *Seema-ko* from the lower position to the higher position to acquire case.

In contrast, the DS analysis, as shown in Figure (1e), does not represent the relationship between *Seema* and *bewakuuf (‘stupid’)* structurally: *Seema* is not a dependent on *bewakuuf*; Instead both *Seema* and *bewakuuf* are dependents of the matrix predicate *samjhaa (‘consider’)*. The relationship between *Seema* and *bewakuuf* is encoded into their dependency labels: *Seema* has the label *k2* and *bewakuuf* the label *k2s*. The (*k2, k2s*) pair indicates that semantically the *k2* node is dependent on the *k2s* node and the dependency relation between them is *k1*.

5.1.2 Handling incompatibility by introducing DS+

When a linguistic phenomenon (e.g., argument sharing in an embedded small clause) is represented in both DS and PS but in different ways, the automatic DS-to-PS conversion is still possible if we can automatically create a new DS, let us call it DS+, which is derived from the original DS but is consistent with the PS. That is, DS+ and PS represent that phenomenon in the same way. For instance, the DS in Figure (1e) is not consistent with the PS in Figure (1c), but the DS in Figure (1f) is because it encodes the sharing aspect of small clause as two coindexed nodes just like in the PS. Furthermore, from the meaning of the (*k2, k2s*) pair, it is easy to write an *ad-hoc* procedure that generates the DS in Figure (1f) from the DS in Figure (1e) automatically.

Therefore, the incompatibility due to representation difference can be handled by introducing a DS+, and the DS-to-PS conversion can be done in two steps: first, given a DS, DS+ is created automatically from the DS; second, a PS is generated from DS+ by applying a conversion algorithm. Determining the shape of DS+ and writing the DS-to-DS+ procedure require good understanding of the difference between the DS and PS analyses. But note that the DS-to-DS+ procedure is entirely independent of the data tokens we are trying to convert; we only need to understand the different representations for the type of phenomenon.

5.2 Phenomena represented only in one side

The DS and PS guidelines are formal linguistic descriptions, but they need not be a complete description of the language. The designers of the treebank may choose not to represent certain linguistic information for practical reasons. For example, the English Penn Treebank does not represent the syntactic structure of prenominal nominal and adjectival modifiers, even though it is generally assumed that such structure exists. Consequently, there could be certain phenomena that are represented in either the DS or PS analyses, but not in both. In the HUTB, one such case is the phenomenon of the unaccusativity/unergativity distinction.

5.2.1 Unaccusativity/unergativity

The unaccusativity/unergativity distinction refers to the fact that intransitive verbs cross-linguistically do not form a unified class - they break down into two classes: unaccusative verbs in which, roughly speaking, the sole argument is semantically a patient (e.g., *open, break*), and unergative verbs in which the sole argument is semantically an agent (e.g., *dance, laugh*). Two examples in Hindi are given in (2) and (3). This meaning difference correlates with a number of syntactic differences and many linguistic theories appeal to the unaccusative/unergative distinction to explain these differences. Other linguistic theories, however, do not make a distinction between these two classes.

(2) Unaccusatives:

darwaazaa	khul	rahaa	hai
door.M	open	Prog.MSg	be.Prs.Sg

‘The door is opening.’

(3) Unergatives:

Ravi	naac	rahaa	hai
Ravi.M	dance	Prog.MSg	be.Prs.Sg

‘Ravi is dancing.’

In the HUTB, the DS guidelines do not make the distinction and the sole argument of both unaccusative and unergative verbs is annotated as *k1*, as shown in Figure 7.

The PS guidelines assume that particular semantic relations (such as patient) are associated with designated structural configurations. Hence, the sole argument of an unaccusative verb needs to combine with the verb in the same position as canonical objects would (as a sister of V). But

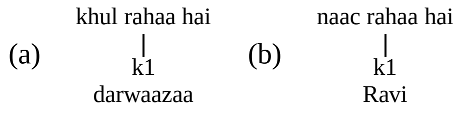

Figure 7: DS for the sentences in (2) and (3)

since the argument also functions as the subject, it must also occupy the position occupied by canonical subjects (sister of V'). This is accomplished in the PS by inserting a special trace in the object position (**CASE**), representing the fact that semantically, the constituent in subject position originates in the object position. The PS analysis follows the standard analysis of unaccusativity as articulated in (Burzio, 1986), and the tree for (2) is shown in Figure 8. In contrast, the sole argument of an unergative verb semantically behaves like an agent and functions as the subject, so it occupies the subject position, as in Figure 9, and there is no object position or the movement from the object position to the subject position. It is easy to show that the (DS, PS) tree pair for the unergative sentence (3) is consistent, whereas the pair for the unaccusative sentence (2) is not.

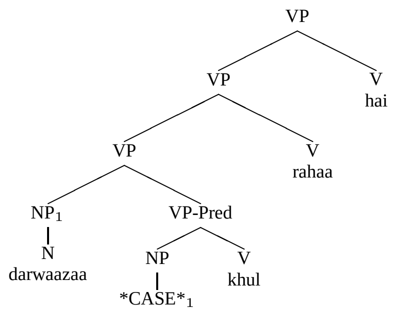

Figure 8: PS for the unaccusative in (2)

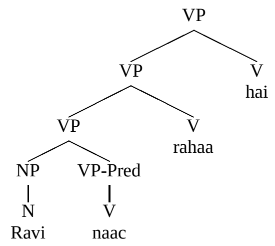

Figure 9: PS for the unergative in (3)

5.2.2 Handling incompatibility requires additional resource

How can we handle the problem that the DS and PS analyses for unergative verbs are compatible, while the ones for unaccusative verbs are not? While one could propose to create a DS+ for unaccusatives like what is done for small clauses, the problem is that this is a property of a data token, and not of the phenomenon of intransitive verbs. We cannot simply use knowledge about this type of phenomenon, since we need to know properties of the particular data token. Because the unaccusative/unergative distinction is not made in the DS, DS+ cannot be created automatically from DS without resorting to an additional resource that will explain the data token. In this case, a list of unergative and unaccusative verbs in Hindi can provide this information, since all instances of a particular intransitive verb are always either unergative or unaccusative. In other words, automatic DS-to-PS conversion is impossible unless an additional resource is provided that allows the conversion mechanism to make the unaccusative/unergative distinction. In the HUTB, the PropBank turns out to be such a resource as it makes the relevant distinction for independent reasons and this allows automatic conversion to proceed.

6 Conclusion

This paper has addressed the issue of when a treebank can be automatically converted to another. We have discussed several important concepts in a treebank and defined compatibility between analyses and consistency between syntactic structures (DS and PS). We have then provided a procedure for comparing treebanks guidelines with respect to conversion. Specifically, we have argued that the conversion from one treebank to another must be examined on a phenomenon-by-phenomenon basis, and that for each phenomenon, there are three scenarios that may arise: the two guidelines have compatible analyses; they have incompatible analyses; and one represents the phenomenon but the other does not. In the first case, automatic conversion is fairly direct; in the second case, we need to study the phenomenon and the analyses proposed for it and provide an intermediate representation to bring the gap; in the third case, we need additional information to achieve the conversion.

Acknowledgments This work is supported by NSF grants CNS-0751089, CNS-0751171, and CNS-0751213. We would like to thank the anonymous reviewers for helpful comments and our colleagues on the Hindi-Urdu Treebank Project for their support.

References

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for indian languages. In *Proceedings of The Third International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing – A Paninian Perspective*. Prentice-Hall of India.

Luigi Burzio. 1986. *Italian syntax: a government-binding approach*. Studies in natural language and linguistic theory. Kluwer, Dordrecht.

Aoife Cahill, Mairead 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*.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings for the 37th Annual Meeting of the Association for Computational Linguistics (ACL-1999)*, pages 505–512. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT-2002)*, San Diego, CA.

Joakim Nivre. 2003. Theory-supporting treebanks. In *In Proceedings of the TLT 2003 Workshop*.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In *Proceedings of ICON-2009: 7th International Conference on Natural Language Processing*, Hyderabad.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations. In *Proceedings of NAACL-2010*.

Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *Proc. of the Human Language Technology Conference (HLT-2001)*, San Diego, CA.

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer, and Dipti Misra Sharma. 2009. Towards a multi-representational treebank. In *The 7th International Workshop on Treebanks and Linguistic Theories (TLT-7)*, Groningen, Netherlands.