Converting SynTagRus Dependency Treebank into Penn Treebank Style

Alex Luu, Sophia A. Malamud, Nianwen Xue
Brandeis University
415 South Street, Waltham, Massachusetts USA
alexluu,smalamud,xuen@brandeis.edu

Abstract

This paper presents the conversion of SynTagRus dependency structures into Penn Treebank style phrase structures, whose resulting data will be used to train a statistical constituency parser for Russian and create a large-scale constituency-parsed corpus. The implemented conversion includes various innovative features in order to create phrase structure trees that are closest to Penn Treebank style while optimally preserving information of the original dependency structure annotations. We believe the newly converted phrase structure treebank will be not only an adequate training dataset for our ongoing project but also a valuable resource for traditional and computational linguistic research.

1 Introduction

A treebank is usually created based on either dependency structure (DS) or phrase structure (PS) such that the selected formalism is optimally compatible with the language under consideration. From this perspective, DS formalism is suited for SynTagRus, the first general-purpose treebank (1M words) for Russian, a Slavic language with a relatively free word order (Boguslavsky et al., 2002). In contrast, existing gold standard corpora involving language variation and change such as Penn corpora of historical English (Kroch and Taylor, 2000; Kroch et al., 2004; Kroch et al., 2016) and the corpus of Appalachian English (Tortora et al., in progress) use PS formalism similar to English Penn Treebank (PTB) (Bies et al., 1995). To facilitate the creation of comparable corpora for less-configurational languages, and to enable the use of the wealth of NLP and theoretical research tools, such as CorpusSearch[1] developed for PTB-style corpora, we aim to enrich this formalism to optimally capture the grammatical details of a free word order language like Russian, and to convert SynTagRus DS into this enriched PTB style PS (henceforth, DS-to-PS conversion[2]) without loss of information. Eventually, we will use the newly converted data to train a statistical PS parser for Russian and create a large-scale PS-parsed corpus. In this paper, we report our effort in developing the enriched PS representation and implementing DS-to-PS conversion.

2 Related Work

To the best of our knowledge, Avgustinova and Zhang (2010) is the only prior work addressing the conversion of SynTagRus DS into PS. Within the framework of Head-driven Phrase Structure Grammar (HPSG) the conversion implemented in this work outputs HPSG-conform PS trees via three steps: converting DS into pseudo PS by creating additional constituent nodes that immediately dominate head words and their dependents, annotating the branches of the pseudo PS with HPSG-oriented schemata, and binarizing the pseudo PS. This conversion process is specific to HPSG framework and cannot be straightforwardly manipulated for PTB style PS. Consequently, we follow a more universal DS-to-PS conversion procedure suggested in (Xia, 2008; Bhatt et al., 2011), including the following steps:

1) DS to DS+: removing non-projectivity
2) DS+ to PS+: simple and general conversion
3) PS+ to PS: handling subtleties

In addition, we adopt the approaches of DS+ to PS+ conversion proposed in (Xia and Palmer,

[1] http://corpussearch.sourceforge.net/index.html.

[2] The code for this conversion is available at https://github.com/luutuntin/SynTagRus_DS2PS.

Proceedings of LAW X – The 10th Linguistic Annotation Workshop, pages 16–21,
Berlin, Germany, August 11, 2016.

2001; Xia et al., 2008), which include simple heuristic rules and take language-specific information as input in defining projections for each syntactic category and attachment levels for each head-dependent pair. Compared with other work on DS-to-PS conversion, (Collins et al., 1999; Aldezabal et al., 2008, a.o.), this approach gives us more flexibility to produce PS that are as close to PTB style as possible while preserving information of the original DS annotations.

An innovation in our proposal is that we use functional tags to represent the extremely fine-grained (and open) list of dependency link types in SynTagRus (Boguslavsky et al., 2002), and utilize this information in the projection rules that create the PS representations.

3 SynTagRus DS-to-PS Conversion

The converted SynTagRus includes 66 dependency link types, 49,420 sentences, 708,480 tokens excluding punctuation marks, 38,311 lemmas, and 1,365 phantom nodes, corresponding to the omitted elements in elliptical constructions.

3.1 Phrase Labeling

We constructed the tag set for our target PS treebank (see Table 1), taking into account language-specific information in SynTagRus.

DS (SynTagRus) POS	**DS POS tag (X)**	**PTB phrase label (XP)**
Noun	S	NP
Adjective	A	ADJP
Verb	V	VP
Adverb	ADV	ADVP
Numeral	NUM	QP
Preposition	PR	PP
Conjunction	CONJ	CONJP
Particle	PART	PRT
Exclamation *yes, no*	P	INTJ
Interjection	INTJ	INTJ
uninflected word	NID	NIDP
combining form	COM	*NP modifier*

Table 1: POS tags and phrase labels.

In addition to the phrase labels presented in Table 1, we use two clause labels, *SS* and *SBAR*, corresponding to *S* (simple declarative clause) and *SBAR* (relative/subordinate clause) in PTB, respectively. To handle *wh*-phrases, we assign the *wh*-feature to every word whose lemma belongs to the list of *wh*-lemmas and whose POS tag is not *CONJ*, using functional tag *-WH*.

3.2 DS to DS+

The free word order of Russian causes a large number of non-projective dependency trees in SynTagRus (cf. Bhatt et al. (2011) for Hindi/Urdu). We propose an algorithm (Table 2) that converts non-projective to projective dependency trees, using traces and co-indexation in the form of null elements **NP2P** (see section 3.5 for a converted example). The recursive helper function path(G) in this algorithm generates a specific sequence of all the nodes in a DS graph G such that any dependent node comes before its head. We call this specific order **tree-oriented**.

Input: a non-projective DS graph G

DS Graph nonprojective-to-projective(G)
  for (each edge of head i & dependent j in G)
  if is-nonprojective-edge(G, i, j)
  insert a null element headed by i and co-indexed with j into G
  for (each node i in path(G))
  if (i is a null element)
  get the co-indexed node j
  assign head of j to variable h
  while (is-nonprojective-edge(G, h, j))
  assign head of h to h
  make h the new head of j
  remove edge between j and its old head

Output: a projective DS graph G

Table 2: DS to DS+ conversion.

3.3 DS+ to PS+

To convert projective dependency graphs (DS+) into the preliminary form of PTB style PS trees (PS+), we decompose the conversion of a complete DS+ (corresponding to a sentence) into a series of conversions for each subgraph of a head node and its (immediate) dependents, which we call a **unit subgraph**. When converting a unit subgraph (Fig. 1), we construct a specific head projection chain for each node in the subgraph, taking into account its POS tag and the dependency links (if any) between it and its head as well as its dependent(s)(see more details in sections 3.3.1 and

3.3.2). In the next step, we attach the root of each dependent's projection chains to the corresponding node in the head's projection chain to form a complete representation of the subgraph.

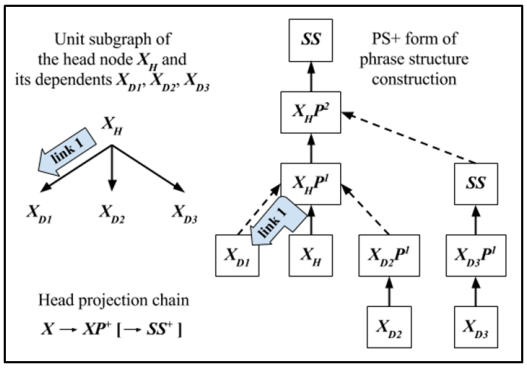

Figure 1: Unit-subgraph DS+ to PS+ conversion.

3.3.1 Head Projection Table

The projection of each node to the phrase level ($X \rightarrow XP$) is defined by the head projection rules (Table 3), based on its POS tag in DS.

DS POS	X → XP
Noun	$S \rightarrow NP$
Adjective	$A \rightarrow ADJP$
Verb	$V \rightarrow VP$
Adverb	$ADV \rightarrow ADVP$
Numeral	$NUM \rightarrow QP$
Preposition	$PR \rightarrow PP$
Conjunction	$CONJ \rightarrow CONJP$
Particle	$PART \rightarrow PRT$
exclamation *yes, no*	$P \rightarrow INTJ$
Interjection	$INTJ \rightarrow INTJ$
uninflected word	$NID \rightarrow NIDP$

Table 3: Head projections at the phrase level.

3.3.2 Link Projection Table

Each syntactic dependency link type, involving a head X_H and a dependent X_D, has its own projection rule. As there are 66 link types in SynTagRus, Table 4 only presents some examples to show the diversity of projection rules that best describe their desired PS construction (e.g. a *relative* link between X_H and X_D will project to X_HP, which is similar to the projection of *link 1* in Figure 1).

Here, we not only reuse as many PTB functional tags (e.g., *PRD* for *predicate* and *SBJ* for *subject*) as possible, but also create new tags that reflect the fine-grained syntactic links in SynTagRus (e.g., *RLT* for *relative* and *EXP* for *expletive*) and therefore are invaluable for implementing different transformations at the PS+ to PS stage.

[3]Tag -PRD is only applied for non-VP predicates

Link type	Link projection
Actant: Predicative	$SS \rightarrow X_HP\text{-}PRD^3\ X_DP\text{-}SBJ$
Attributive: Relative	$X_HP \rightarrow X_H\ \ X_D\text{-}RLT$
Coordinative	$X_HP\text{-}CRD \rightarrow X_HP\ \ X_DP$
Auxiliary: Expletive	$X_HP \rightarrow X_H\ \ X_DP\text{-}EXP$

Table 4: Link projections.

Our treatment of the differentiation between sister and Chomsky adjunction departs from Xia and Palmer (2001) and is similar to the optimization implemented in Xia et al. (2008). This differentiation is needed to produce PS that are close to the trees in PTB. To treat each type of dependency link in SynTagRus appropriately, we directly incorporate the concrete adjunction styles into the projection rules for each link, rather than distinguishing them in the modification table and implementing an additional step to handle Chomsky adjunction structures, as Xia and Palmer (2001) do. For example, in Table 4 the *coordinative* link corresponds to Chomsky adjunction (in which the head node necessarily projects to the phrase level) while the *expletive* link corresponds to sister adjunction (in which the head node does not necessarily project to the phrase level).

3.3.3 Construction of PS+ Trees

We use the algorithm presented in Table 5 for converting DS+ into PS+.

Input: a projective DS graph *G*

Tree DS-toPS(*G*)
  for (each node *i* in *G*)
  get projection chain *c* of *i*
  build (non-branching) PS tree for *i* using *c*
  for (each node *i* in path(*G*))
  attach dependents' PS trees to *i*'s PS tree
  return PS tree *T* of the root node

Output: a PS tree *T*

Table 5: DS+ to PS+ conversion.

In order to preserve the linear word order of all nodes in a unit subgraph, the projection chain of the dependent which is linearly farther from the head should not be attached to a lower position

in the projection chain of the head. If this violation occurs, we will move up the attachment position of this dependent chain until it is at least equal to those of the dependents which are linearly closer to the head. In other words, we attach it as low as possible as long as this does not cause non-projectivity. Additionally, we insert a null element *, co-indexed with the moved-up node, in the original attachment position of the moved-up node. This null element as well as the null element *NP2P*, which is introduced when eliminating non-projective trees, are descriptive devices for capturing scrambling phenomena in Russian in a theory-neutral manner.

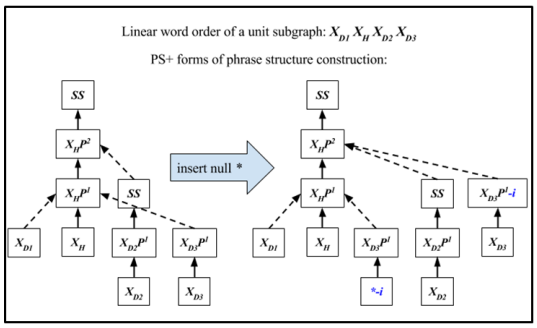

Figure 2: Insertion of null element *.

3.4 PS+ to PS

In this stage we implement the following types of tree transformations:

1) Label replacement, e.g. changing *CONJP* to *SBAR* for subordinative structures

2) *Wh*-movement, e.g. adding null elements *T*, *SBAR* nodes for relative clauses

3) Eliminating intermediate nodes, so that in phrase structure trees, the dependents in formerly non-projective edges c-command their traces null elements *NP2P*

4) Label merging, mainly used for handling coordinative structures

It is worth emphasizing that the resulting PS in PTB style adequately preserve all the enriched information of SynTagRus DS annotation.

3.5 A Converted Example

We examine the sentence in Figure 3, involving several phenomena characteristic of Russian: an impersonal modal *nado* "its necessary" which takes an infinitival phrase as its argument, a scrambled accusative object of the infinitive *knopku* "button.ACC" which participates in

Кнопку надо нажимать той же
Knopku nado nažimat' toj že
Button.ACC necessary press.INST that.INF PRT

рукой, которой делается ход.
rukoj, kotoroj delaetsja xod.
hand.INST, which.INST make.3S.SPASS move.NOM.

It's necessary to press the button with the same hand with which the move is made.

Figure 3: An example sentence.

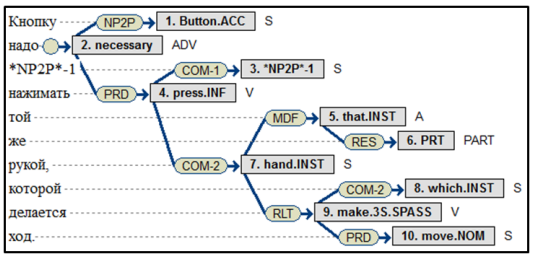

Figure 4: Non-projective SynTagRus DS.

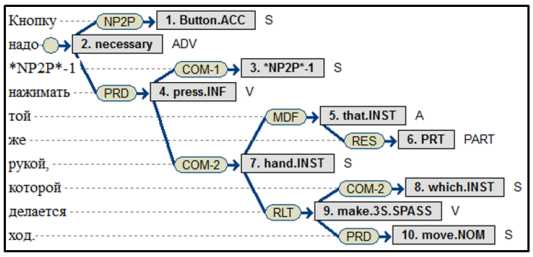

Figure 5: Projective SynTagRus DS+.

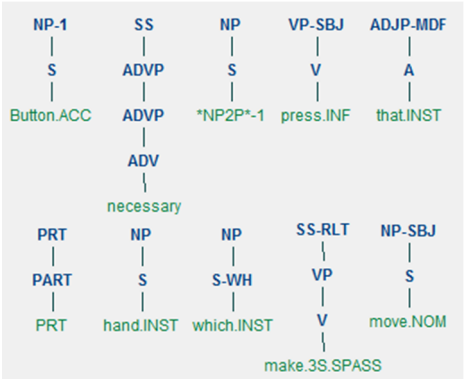

Figure 6: Non-branching PS trees.

a non-projective dependency, and a relative clause containing a *sja*-passive. The original DS of this sentence in SynTagRus, presented in Figure 4, includes the non-projective edge of syntactic link *COM-1* (i.e. *1st completive*) between "button.ACC" and "press.INF". This non-projectivity is resolved by the DS to DS+

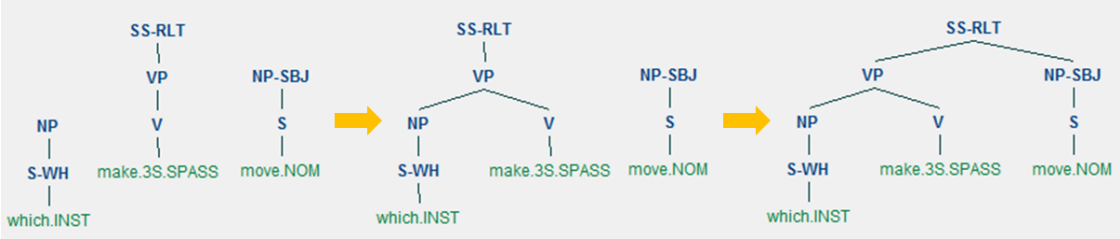

Figure 7: PS+ construction for a unit subgraph.

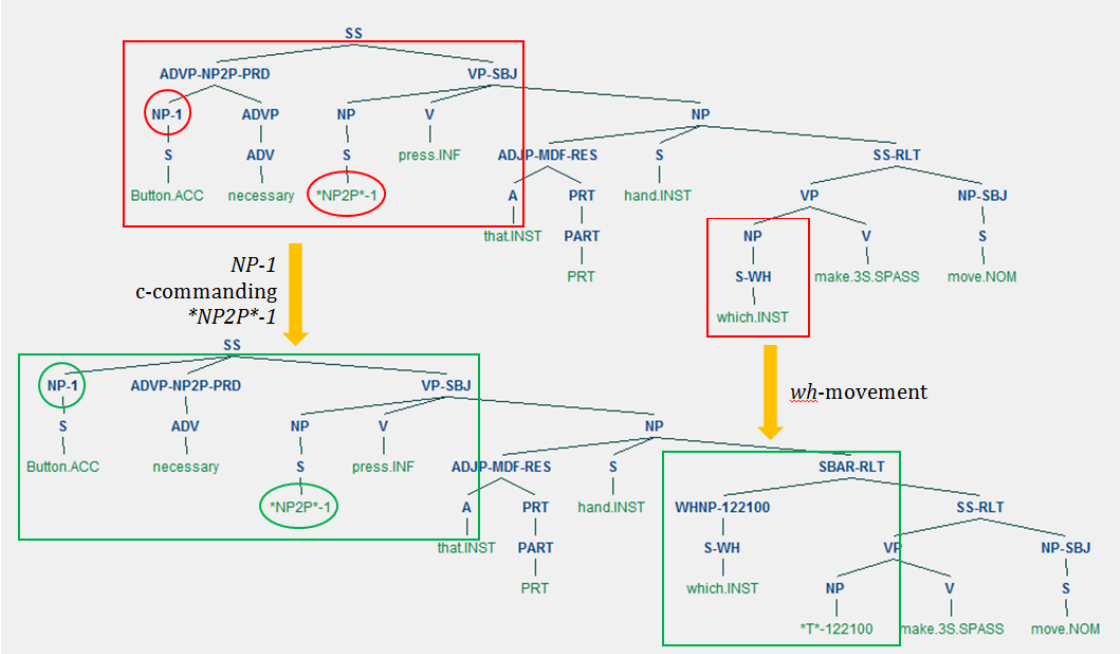

Figure 8: PS+ to PS conversion.

conversion, whose output DS+ is shown in Figure 5. Specifically, "button.ACC" is moved up to attach to "necessary" via general link *NP2P*; meanwhile, null element **NP2P*-1*, co-indexed with "button.ACC" (the first node in DS), is inserted between "necessary" and "press.INF", occupying the original position of "button.ACC" in DS. To create PS+, we first build the non-branching PS trees for all nodes in DS+ (Fig. 6). Next, we construct a PS tree for every unit subgraph in DS+ according to the following tree-oriented order: "move.NOM" → **NP2P*-1* → "PRT" → "that.INST" → "which.INST" → "make.3S.SPASS" → "hand.INST" → "press.INF" → "button.ACC" → "necessary". For example, the construction of PS+ for the unit subgraph headed by "make.3S.SPASS" is presented in Figure 7. Finally, Figure 8 shows the conversion from the PS+, the upper tree, to the PS, the lower tree, which involves such transformations as scrambled constituents c-commanding their traces and *wh*-movement.

4 Conclusions and Future Work

In this paper, we report on a conversion of the SynTagRus DS corpus into PTB style PS, preserving the information contained in the original DS annotations. We are currently working to refine our PS annotation guidelines and manually correct the converted data to create the gold standard for evaluating the implemented conversion. After this evaluation, the newly converted corpus will be distributed under the same noncommercial license as SynTagRus in its original form. We believe that the resulting PS treebank and the enriched PS formalism will be not only an adequate training dataset for automatic parsing of new Russian data, but also a valuable resource for traditional and computational linguistic research.

Acknowledgments

We would like to thank Marie Meteer and the anonymous reviewers for very constructive comments and valuable suggestions. All errors and mistakes are, of course, the responsibility of the authors.

References

Izaskun Aldezabal, Maria Jesùs Aranzabe, Arantza Diaz de Ilarraza, and Enrique Fernández. 2008. From dependencies to constituents in the reference corpus for the processing of Basque. *Procesamiento del Lenguaje Natural*, 41:147–154.

Tania Avgustinova and Yi Zhang. 2010. Conversion of a Russian dependency treebank into HPSG derivations. In *Ninth International Workshop on Treebanks and Linguistic Theories*, page 7.

Rajesh Bhatt, Owen Rambow, and Fei Xia. 2011. Linguistic phenomena, analyses, and representations: Understanding conversion between treebanks. In *IJCNLP*, pages 1234–1242. Citeseer.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. *University of Pennsylvania*, 97:100.

Igor Boguslavsky, Ivan Chardin, Svetlana Grigorieva, Nikolai Grigoriev, Leonid L Iomdin, Leonid Kreidlin, and Nadezhda Frid. 2002. Development of a dependency treebank for Russian and its possible applications in NLP. In *LREC*. Citeseer.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512. Association for Computational Linguistics.

Anthony Kroch and Ann Taylor. 2000. The Penn-Helsinki parsed corpus of Middle English (PPCME2). *Department of Linguistics, University of Pennsylvania, CD-ROM*.

Anthony Kroch, Beatrice Santorini, and Lauren Delfs. 2004. The Penn-Helsinki parsed corpus of early modern English (PPCEME). *Department of Linguistics, University of Pennsylvania, CD-ROM*.

Anthony Kroch, Beatrice Santorini, and Ariel Diertani. 2016. The penn parsed corpus of modern British English (PPCMBE2). *Department of Linguistics, University of Pennsylvania, CD-ROM*.

Christina Tortora, Beatrice Santorini, and Frances Blanchette. The audio-aligned and parsed corpus of Appalachian English (AAPCAppE). In progress.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics.

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer, and Dipti Misra Sharma. 2008. Towards a multi-representational treebank. *LOT Occasional Series*, 12:159–170.

Fia Xia. 2008. General techniques for creating treebanks. Lectures of TCS NLP Winter School, collocated with the Third International Joint Conference on Natural Language Processing, Hyderabad, India.