ProppML: A Complete Annotation Scheme for Proppian Morphologies

W. Victor H. Yarlott[1] and Mark A. Finlayson[2]

1 **School of Computing & Information Sciences, Florida International University, Miami, FL, USA**
wvyar@cs.fiu.edu

2 **School of Computing & Information Sciences, Florida International University, Miami, FL, USA**
markaf@fiu.edu

Abstract

We give a preliminary description of ProppML, an annotation scheme designed to capture all the components of a Proppian-style morphological analysis of narratives. This work represents the first fully complete annotation scheme for Proppian morphologies, going beyond previous annotation schemes such as *PftML*, ProppOnto, Bod *et al.*, and our own prior work. Using ProppML we have annotated Propp's morphology on fifteen tales (18,862 words) drawn from his original corpus of Russian folktales. This is a significantly larger set of data than annotated in previous studies. This pilot corpus was constructed via double annotation by two highly trained annotators, whose annotations were then combined after discussion with a third highly trained adjudicator, resulting in gold standard data which is appropriate for training machine learning algorithms. Agreement measures calculated between both annotators show very good agreement ($F_1 > 0.75$, $\kappa > 0.9$ for functions; $F_1 > 0.6$ for moves; and $F_1 > 0.8$, $\kappa > 0.6$ for dramatis personae). This is the first robust demonstration of reliable annotation of Propp's system.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Narrative structure, Computational folkloristics, Russian folktales

Digital Object Identifier 10.4230/OASIcs.CMN.2016.8

1 Introduction

Vladimir Propp's *Morphology of the Folktale* is Propp's attempt to "make an examination of the forms of the tale which will be as exact as the morphology of organic formations" [39, p. xxv]. While previous work on motifs and tale types by Roman Volkov [44] and Antti Aarne [1] made attempts at creating a system for expressing and describing components of folktale, Propp dismissed these as both unscientific and suggesting the incorrect notion that there is a clear-cut division into types. As the first example of its kind, Propp's work aimed to capture the formulaic repetition that is present in folktales in a precise and relatively formal way.

With today's more advanced mathematical and conceptual machinery, Propp's theory can be described as a *plot grammar*, which lists the elemental plot pieces and their possible orders that may occur in folktales (Propp called his plot pieces *functions*). Propp's theory also describes the high-level organization of tales (*moves*), the instantiated forms of functions (what we have called elsewhere *function subtypes*), long-distance dependencies between function subtypes, exceptions and other complications (*inversions* and *trebling*), and character archetypes (*dramatis personae*), which correlate with particular functions. To derive and

provide evidence for his theory he analyzed approximately 100 Russian hero tales drawn from the collection of Aleksandr Afanas'ev [2], providing for each tale a list of functions that appear and the order of their combination. Propp noted that his particular list of functions and subtypes, and their orders, was applicable only to this specific set of Russian tales; others, such as Alan Dundes [8] and Benjamin Colby [6], have shown, however, that Proppian-style analyses may profitably be performed for other cultures.

Propp's morphology is a seminal work in the field of folklore and narratology, inspiring many counter-reactions [31, 32] and extensions [8, 25], and it continues to inspire generations of scholars. Within the field of computational narrative, Propp's morphology has a substantial number of potential applications for three reasons. First, Propp's morphology is one of the most formal narratological treatments developed so far, having a relatively clear method for determining and extracting theory components from text. Second, Propp's work separates content from form, allowing a description and analysis of a plot without requiring its instantiation directly into language. These two points combine to make Propp's morphology readily applicable to the creation of computational models. Third, properties of Propp's morphology—that functions always occur in the same sequence—makes them a powerful tool for story generation. Propp's work has been applied in systems as diverse as textual story generation [12, 23, 24], support and guidance for children during story creation activities [33], and as a means of varying sign language for virtual characters based on conflict to enhance computer generated sign language [41].

There have been a number of objections to Propp. Propp's work has been criticized for its separation of form from content, treating content as "less important" than form [32, p. 179], making the mistake of trying to "characterize a tale without mentioning the motifs" [5, p. 194]. His work has been considered inconsistent, because even as he attempts to separate content from form, morphological criteria "reintroduce some aspects of content" [32, p. 179]. Additional criticisms target the reproducibility of Propp's work [4] and that Propp's work is not sufficient to account for the diversity of plots in fairy tales [5].

Surprisingly, despite the deep interest in Propp's work in computational narrative circles, one of the most fundamental questions regarding Propp's morphology has not been satisfactorily answered: taking Propp's list of functions as a given, is the morphology specified precisely enough that an independent person will find, in the same tales, the same functions that Propp did?

In this work we address this question. We designed a complete annotation scheme for Propp's morphology, which goes beyond all previous annotation schemes to capture every piece of Propp's morphology in a linguistically sophisticated way: past work (e.g., [35]) does not address implicit functions, dramatis personae, or moves, nor does it provide a fully annotated corpus. We ran a double-annotation experiment where we extensively trained three people over several months in the workings of Propp's system (especially the list of functions and subtypes), and had them annotate 15 single-move tales drawn from Propp's original corpus. It should be noted that while these tales are all technically single-move, in accordance with Propp's original annotations, preparatory functions are not a part of the main move and thus comprise, in effect, their own 'preparatory' move. Agreement measures calculated over this set of data shows that Propp's system can be reliably applied by human annotators to single-move tales. While prior work has addressed this question (e.g., [4]), prior work has failed to adequately train annotators, our work addresses this shortcoming and is the first to robustly demonstrate the reliability of applying Propp's system. This result shows that human annotation of Propp's morphology on single-move tales has a high agreement ($F_1 > 0.75$ and $\kappa > 0.9$ for functions, $F_1 > 0.6$ for moves, $F_1 > 0.8$ and $\kappa > 0.6$

for dramatis personae). In general, these results are very good, with the κ for functions being "near perfect." Annotation and analysis of multi-move tales is ongoing, but show the same trends and will be reported in future articles.

Our work lays the foundation for the machine learning of Proppian functions. Preliminary versions of the data presented here [13] has already been used to demonstrate that learning Propp's function from text is possible [16].

1.1 Outline of the Paper

In §2, we describe how the experimental design of double-annotation addresses the question at hand. Then, in §3, we explain how our annotation scheme improves on prior work to design of our annotation scheme, and give a high-level description of the scheme with an example annotation. We give a full formal BNF specification of the scheme in Appendix A. In §4, we describe the selection of data, the annotation procedure (including training of the annotators), and the tools used during annotation. In §5 we show the results and discuss the agreement measures. In §6 we cover related work, including prior attempts at answering the experimental question.

2 Experimental Design

The question at hand is whether Propp's theory is *reliable*. By *reliable*, we mean something quite specific: will independent people agree with each other when applying Propp's theory? There are several ways to define "agree with each other," in particular:

Q1. Taking as given Propp's general approach, list of functions, and identified functions in specific tales, will independent people agree with each other as to where and whether those functions appear in those tales?

Q2. Taking as given Propp's general approach and list of functions, will independent people agree with each other, and also agree with Propp (as appropriate), when asked to find his functions in tales?

Q3. Taking as given Propp's general approach, will independent people agree with each other, and with Propp (as appropriate), as to the set of functions that are indicated by a particular set of tales?

As can be seen the level of generality of the questions progresses from quite specific in Q1 to fairly general in Q3. In this work we address Q1 with regard to single-move tales. We will address Q1 for multi-move tales and Q2 generally in future work. As for Q3, others have addressed this manually [9, 6], and we have begun to address it computationally [16].

What experimental design is appropriate to answer Q1? As pointed out by Bod *et al.* [4], a double-annotation paradigm is one appropriate approach to addressing the reliability of a textual marking scheme, and is the approach we follow here. In a double-annotation experiment, two people are trained in the operation of the scheme (these are called the *annotators*), and are asked to independently mark up texts with the scheme. The agreement between the two sets of markings created by the annotators is measured using appropriate statistical agreement measures such as the F-measure [20, 43] or Fleiss' kappa [19]. High agreements indicate a positive answer to the question. Further, a gold standard marking of the texts (suitable for machine learning) can be generated by having the two annotators confer to resolve disagreements, sometimes assisted by a third party (called the *adjudicator*).

Conducting such a double-annotation experiment entails the following steps:

1. Define an appropriate and complete annotation scheme. (§3)
2. Select the texts to be annotated. (§4.1)
3. Assemble or build the tools required to do the annotation. (§4.2)
4. Train the annotators and adjudicator in both the scheme and the tools. (§4.3)
5. The annotators perform the annotation. (§4.4)
6. Measure agreement between the annotators. (§5)
7. *Optional:* The adjudicator eliminates disagreements to generate gold standard data. (§4.4)

In reality, there is often a loop between steps 6 and 1, as noted elsewhere [40, 17], because analysis of the data reveals flaws in the scheme, which requires revision and a repeat of the experiment. We had already progressed through such a loop several times during previous attempts at annotation of Propp's scheme, and we discuss the lessons we learned in those loops, and how they were integrated into this final scheme, in the next section.

3 Design of ProppML

The point of an annotation scheme is to capture, in a precise way, all the different "moving parts" of the phenomena to be annotated. There should be a way of notating every important distinction provided by the theory that backs the scheme, and, for text annotation schemes, associating those notations to the relevant spans of text.

In our approach to annotating Propp's morphology, we split the task into three separate schemes: functions, moves, and dramatis personae. Collectively we refer to these as *ProppML*. Although the schemes are separate, they do cross-reference each other in specific places (i.e., the move scheme refers to the function scheme), as well as reference other related annotation schemes as described below.

All three schemes allow association of Propp's theoretical constructs of functions, moves, and dramatis personae with the text under consideration. This association is implemented by reference to character offsets, anchored by identified token boundaries. As described in [13], the texts are first run through a tokenizer (in this case, from the Stanford CoreNLP suite [36]) that splits the text into single tokens. These tokens are indexed to character offsets in the text as described in Appendix A. The ProppML schemes then refer to the tokens by unique id numbers.

We sketch out the important parts of ProppML in each of the three following subsections. The full details for each of these parts are found in Appendix A.

3.1 ProppML: Function Scheme

The first of the three schemes allows an annotator to notate the presence of functions in the text. Functions are split into two portions: the **function tag** and one or more function **instances**. The function tag allows the annotator to mark three pieces of information: the **type** of the function, whether or not the function is **inverted**, and the **symbol** for the function.

The function type marks the function as a *Normal*, a *Prepatory* function, or the *Initial* situation. Separating these out is important because Propp does not specifically note where the Preparatory and Initial functions occur, and preparatory and initial functions do not participate in the normal Move structure of the tale [39, pp. 108–109].

The primary symbol of the function is one of the most memorable parts of Propp's theory. He identified, for example, the function *Villainy*, and gave it the symbol *A*. Propp gave

the function *Reward* the symbol W. Importantly, Propp also identified various function subtypes, indicated by a combination of superscripts and subscripts on the primary symbol. Our scheme allows the symbol and its super- and sub-scripts to be marked by alphanumeric strings in the appropriate positions.

One of the important innovations of our marking scheme is the notion of a **function instance**. In our early approaches to marking Propp [13], when Propp indicated a function was present, we noted that there often was not one unambiguous occurrence of that function. Propp himself admited this when he identified the phenomenon of *trebling*, which is where a particular function is repeated three times. We found that repetition of this sort occurred in many places where Propp did not indicate it. Our scheme allows these occurrences to be marked as separate "instances" of the same function, which is important because instances of one function can interleave with instances of other functions when the functions are repeated in sequence. For example, in The Magic Swan Geese, the little girl encounters three potential helpers (the stove, the tree, the river) at three separate times both when chasing the swan geese and fleeing them[1]. In each case the *DEF* function sequence happens three times, and in each repetition an instance of D precedes an instance of E which precedes an instance of F. Therefore, without the ability to separate these instances, the scheme looses fidelity to Propp's theory.

Annotators may also mark a function instance as **inverted**, which was a part of Propp's theory which indicated when a plot piece did not fulfill its intended plot function, or was fulfilled by a semantically opposite form [39, p. 116, note 4].

For each function instance the annotator is required to mark the **type**. This is an important innovation of our scheme, because often Propp indicates that a function is present in the text, but there is no span of text directly corresponding to the function. For example, in Tale 148, Nikita the Tanner, the function B (dispatch of the hero on the quest) appears explicitly when the Tzar asks Nikita to fight the dragon and rescue his daughter. In these cases the annotator marks the instance with the type *Actual*, which indicates that the instance actually occurs in the text of the tale. In the next sentence we see Nikita preparing to fight, and so clearly the function C (the decision to go on the quest), has already occurred. In these cases the annotators are instructed to find the closest logically connected event, mark that as the instance, then mark the instance as either *Antecedent* or *Subsequent*, depending on whether the connected event is before or after the presumed occurrence of the function in the timeline.

Finally, annotators are also able to mark text spans corresponding to the *signals* for the function and any inversion, and the full extent of the function. The separation of spans into a *signal* portion and an *extent* is important. The identification of a signal span allows the annotators to mark the key verbs (or other words) that most strongly indicate the presence of the function or the inversion. The extent, on the other hand, allows the annotators to indicate the full portion of the text that represents the function. The details of these spans are covered in the Appendix.

3.2 ProppML: Move Scheme

The second of the three schemes allows annotators to mark the Move structure of the tale. Moves are defined as the development from villainy or lack (functions A or a) to a terminal

[1] Propp calls this "trebling", although in practice this sort of repetition occurs in our corpus anywhere from two to four times

function; each lack or act of villainy creates a new move [39, p. 92]. The representation of the move in our scheme is rather simple. Annotators first mark the *move number*, which is a non-negative integer. The move numbered zero is conventionally deemed the "preparatory" move, meaning all the preparatory functions, as well as the initial situation, are contained within it. Thus even "single-move" tales in our scheme can have two moves: the preparatory "pseudo-move", and the actual move.

Each move is then represented as a sequence of function *instances*. Importantly, moves are sets of function instances rather than functions themselves, as different instances of a repeated function can be spread across different moves (see, for example, Tale 93 in [39, pp. 136–137]). Within a move, function instances are ordered by their appearance in the text.

3.3 ProppML: Dramatis Personae Scheme

The final annotation scheme in ProppML is the dramatis personae scheme, which marks the role of various characters. As Propp describes, there are seven different character roles: Hero, Villain, Donor, Helper, Princess, Dispatcher, and False Hero [39, pp. 79–80]. In the data presented below, all the tales were previously marked with "coreference groups" corresponding to bundles of co-refering referential expressions [13]. For example, in the text, a single character might be referred to at different times as "Nikita", "Nikita the Tanner", "he", "him", or "a tanner in the city of Kiev". All these referential expressions referring to Nikita are marked and bundled together into a single coreference group that is assigned a unique identifying number. Annotators are then asked to assign any number of the seven labels (including none) to each co-reference group in the text. Allowing multiple labels to be assigned to a single character is important, as Propp notes that a single character can fulfill multiple different roles, or a single role may be spread among different characters [39, pp. 80–81].

3.4 Example Annotations

Figure 1 shows excerpts from actual annotated files, with the function, move, and dramatis personae annotations. These files are in Story Workbench annotation format, which is described elsewhere [14, 15]. A detailed BNF for each of the three annotation schemes is found in Appendix A.

4 Data Production

4.1 Selection of Texts

Because answering Q1 requires us to know Propp's list of functions for a tale, our raw text is necessarily drawn from Propp's original corpus, which he selected from Aleksandr Afanas'ev's collection of Russian folktales [2]. Propp analyzed 100 of these tales, publishing a subset of his analyses (45 tales) in a table at the end of his monograph [39].

For this pilot study we restricted our analysis to single-move tales. The primary reason for this is that the annotation is time-consuming, and the single move tales are among the shortest in the collection. Selecting the single-move tales facilitated the production of a pilot study, as well as provided initial evidence that further effort applied to the multi-move tales (a much larger and longer set of text), would not be a waste. Further, we have already deeply annotated a set of fifteen single-move tales (18,862 words) for our other work [13], and it seemed appropriate to begin with those as the translations were already available.

```
<?xml version="1.0" encoding="UTF-8"?>
<story>
  <rep id="edu.mit.story.char">
    <desc id="0" len="4064" off="0">
    ...
    An old man lived with his old wife; they had a daughter and a little son. "Daughter, daughter," said the
    mother, "we are going to work; we shall bring you back a bun, sew you a dress, and buy you a kerchief. Be
    careful, watch over your little brother, do not leave the house." The parents went away and the daughter what
    they had told her; she put her brother on the grass beneath the window, ran out into the street, and became
    absorbed in games. Some magic swan geese came, seized the little boy, and carried him off on their.
    ...
    </desc>
  </rep>
  <rep id="edu.mit.parsing.token">
    <desc id="3" len="2" off="353">An</desc>
    <desc id="4" len="3" off="356">old</desc>
    <desc id="5" len="3" off="360">man</desc>
    ...
    <desc id="866" len="6" off="4048">father</desc>
    <desc id="867" len="7" off="4055">arrived</desc>
    <desc id="868" len="1" off="4062">.</desc>
  </rep>
  ...
  <rep id="edu.mit.semantics.rep.function" ver="0.5.0">
    <desc id="2555" len="73" off="353">
      INITIAL::alpha:false::|ACTUAL:::3~4~5~6~7~8~9~10~11~12~13~14~15~16~17~18~19~20
    </desc>
    <desc id="2558" len="182" off="450">
      PREPARATORY::gamma:false:1:2|ACTUAL:27::27~28~29~30~31,57~58~59~60~61~62~63~64~65~66~67~68~69~70~71~72
    </desc>
    <desc id="2560" len="186" off="468">
      PREPARATORY::beta:false::1|ACTUAL:34,75::32~33~34~35~36,73~74~75~76
    </desc>
    <desc id="2559" len="107" off="704">
      PREPARATORY::delta:false::|ACTUAL:88,98,105::87~88~89~90~91~92~93~94~95~96~97~98~99~100~101~102~103~104~
      105~106~107~108
    </desc>
    <desc id="2514" len="87" off="814">
      NORMAL::A:false::1|ACTUAL:116,122::110~111~112~113~114~115~116~117~118~119~120~121~122~123~124~125~126~
      127~128
    </desc>
  ...
  </rep>
  <rep id="edu.mit.discourse.rep.coref">
    <desc id="2456" len="3701" off="353">father|2158,2159,...</desc>
    <desc id="2457" len="3668" off="375">mother|2160,2167,...</desc>
    <desc id="2458" len="3665" off="389">parents|2161,2169,2170,2179,2182,...</desc>
    <desc id="2459" len="3538" off="398">the children|2163,...</desc>
    <desc id="2460" len="3638" off="398">daughter|2162,2165,2166,2171,2174,2176,2180,2183,2184,2185,...</desc>
    <desc id="2461" len="3211" off="413">son|2164,2177,2186,2192,2193,...</desc>
    <desc id="2462" len="235" off="467">what they had told her|2168,2181</desc>
    <desc id="2463" len="5" off="515">a bun|2172</desc>
    <desc id="2464" len="7" off="530">a dress|2173</desc>
    <desc id="2465" len="10" off="551">a kerchief|2175</desc>
    <desc id="2466" len="3338" off="621">the house|2178,...</desc>
    <desc id="2467" len="29" off="727">the grass beneath the window|2187</desc>
    <desc id="2468" len="10" off="746">the window|2188</desc>
    <desc id="2469" len="10" off="771">the street|2189</desc>
    <desc id="2470" len="5" off="806">games|2190</desc>
    <desc id="2471" len="3117" off="814">Swan Geese|2191,2194,...</desc>
    <desc id="2472" len="2691" off="889">Geese wings|2195,...</desc>
    ...
  </rep>
  <rep id="edu.mit.semantics.rep.move" ver="0.1.0">
    <desc id="2556" len="458" off="353">0|2555,2558,2560,2559</desc>
    <desc id="2549" len="3015" off="814">1|2514,...</desc>
  </rep>
  <rep id="edu.mit.semantics.rep.archetype" ver="0.1.0">
    <desc id="2529" len="3638" off="398">HERO|2460</desc>
    <desc id="2544" len="3211" off="413">PRINCESS|2461</desc>
    <desc id="2530" len="3117" off="814">VILLAIN|2471</desc>
    ...
  </rep>
</story>
```

Figure 1 Selection of the annotation of The Magic Swan Geese [26, p. 349], provided for illustrative purposes. Ellipses indicate removal of data to improve readability. We have included **six** annotation layers, enclosed in `<rep>` tags: the text itself, tokens, Proppian function annotation, coreference annotation, move annotation, and archetype annotation. For the function, coref, move, and archetype layer, we only include the data applicable to the selection of text in the text layer. An in-depth description of the annotation scheme can be found in Appendix A.

While Propp's work was performed on the original Russian text, all of our work is performed on English translations. This is generally considered acceptable for first-order structural analysis [18]. Even given these restrictions, this pilot corpus is still a significantly larger set of data than annotated in previous studies.

4.2 The Story Workbench

Linguistic annotation usually requires some sort of computer tool support. In our case, we had already developed the Story Workbench, a robust tool for annotating text in multiple representations [14, 15]. The Story Workbench is a cross-platform and highly extensible generic text annotation tool that was used to actually apply the ProppML annotation schemes to the texts. The Story Workbench builds on top of popular, well-supported tools, uses open-source libraries, and adopts widely-used and well-documented standards..

Story Workbench distinguishes between the programmatic format of annotations (*representations*) and the actual annotations in that format (*descriptions*). Representations in Story Workbench are designed to be reused and applied to multiple texts, rather than existing only in annotations. There are currently more than 17 representations implemented in Story Workbench, including the Proppian functions, archetypes, and moves we describe in this paper.

Story Workbench supports single annotation, double annotation, and annotation development. It supports the automatic creation of annotations, annotation problem identification and migration, inspection of annotations, manual creation and correction of annotations, comparing and contrasting annotations, annotation adjudication, and annotation scheme implementation, among other features [13, The Story Workbench].

To perform the experiment described here, we implemented all three ProppML annotation schemes in the Story Workbench codebase. The Story Workbench provides a user interface for manually creating all the annotations described here, and enforces syntactic consistency and well-formedness. The annotation tool, along with documentation and references fully explaining the capabilities and usage of Story Workbench, may be downloaded online[2] and used for other annotation projects involving ProppML.

4.3 Annotator Training

As discussed previously, annotation was done in a double-blind manner by two highly-trained annotators. Both annotators were students at Harvard University in Cambridge, MA[3] To begin we trained the annotators for three weeks where the annotators were first asked to read Propp's book from start to finish, and then asked to annotate the Magic Swan Geese tale, the analysis of which is explained in detail in Propp's book [39, pp. 96–99]. Reading the book and the initial annotation of the Magic Swan Geese took approximately 20 hours total. The annotators were then brought together then with the adjudicator for a three-hour meeting to discuss any questions and compare their annotations. The adjudicator was already highly trained in Propp's system: he was one of the original annotators who helped produce the first set of Propp annotations [13], and was also a Ph.D. student in English literature at Harvard University. After the annotators received feedback on their annotations, they re-did their annotations on the Magic Swan Geese and had another meeting (another 10 hours total). At this point agreement was very good, and annotation of the remainder of the data began.

Annotators were also trained separately in the operation of the Story Workbench, which took approximately one hour.

---

[2] `http://projects.csail.mit.edu/workbench`

[3] The study was begun while Dr. Finlayson was a researcher at MIT, also in Cambridge.

Table 1 Annotation agreement measures for all stories. F_1^s is the strict F_1 measure, F_1^i is the identification F_1 measure, and F_1^g is the grouping F_1 measure. κ_{sym} is the Fleiss Kappa score for primary function symbols and κ_{dp} is the Fleiss Kappa score for dramatis personae markings. The cumulative measures are the microaverages of the scores in the rest of the column.

		Functions			Moves		Dramatis Personae		
#	English Title	F_1^s	F_1^i	κ_{sym}	F_1^s	F_1^g	F_1^s	F_1^i	κ_{dp}
148	Nikita the Tanner	0.80	1.00	0.89	0.00	0.90	1.00	1.00	1.00
113	The Magic Swan Geese	0.33	0.83	0.93	0.00	0.70	0.74	0.93	0.48
163	Bukhtan Bukhtanovich	0.00	0.70	0.66	0.00	0.57	0.67	0.67	0.37
162	The Crystal Mountain	0.41	0.69	1.00	0.50	0.69	0.73	0.73	0.47
151	Sharbarsha the Laborer	0.22	1.00	1.00	0.50	0.72	0.67	0.67	0.35
152	Ivanko the Bear's Son	0.24	0.82	0.67	0.00	0.55	0.86	0.86	0.62
131	Frolka Stay-at-Home	0.40	0.80	1.00	0.50	0.85	0.90	0.90	0.77
108	Ivashko and The Witch	0.29	0.79	1.00	0.00	0.78	0.89	1.00	0.73
145	The Seven Simeons	0.12	0.71	0.80	0.00	0.63	0.95	0.95	0.76
135	Ivan Popyalov	0.10	0.76	1.00	0.00	0.64	0.67	0.71	0.38
149	Serpent & Gypsy	0.10	0.57	0.57	0.00	0.39	0.67	1.00	0.36
114	Prince Danila Govorila	0.21	0.72	0.92	0.00	0.70	0.75	1.00	0.46
127	Merchant's Daughter	0.21	0.71	1.00	0.00	0.63	0.73	1.00	0.50
140	Dawn, Evening, & Midnight	0.33	0.73	0.90	0.00	0.63	0.95	0.95	0.88
154	Runaway Soldier & the Devil	0.27	0.73	1.00	0.00	0.71	0.89	0.86	0.76
Cumulative		0.27	**0.76**	**0.92**	0.11	**0.67**	**0.82**	**0.88**	**0.63**

4.4 Annotation Procedure

Annotation of the texts after the Magic Swan Geese was performed at a rate of approximately 2,000 words/week, with annotators spending approximately 7 hours a week annotating, and 3 hours/week in an adjudication meeting. Therefore, after the initial training period of 30 hours, the annotators spent approximately 9 weeks annotating, and each annotator spent a total of approximately 120 hours on the project. The adjudicator spent approximately 24 hours on the project, not counting the annotation that he performed in previous years (which constituted, at a minimum, approximately 100 hours of work). During each adjudication meeting disagreements between the annotators were discussed, and additional discussion of subtleties of Propp's system was held as needed. Further, a gold standard marking was produced by the team. Thus, the project produced three sets of marked texts: one marked by annotator 1, another marked by annotator 2, and a gold-standard set corrected by the adjudicator. If the team had a disagreement that could not be resolved, they consulted the project manager[4].

5 Agreement Results

The agreement between the annotators was assessed for each of functions, moves, and dramatis personae in several ways.

For functions, we report two F_1 measures: a 'strict' F_1 which requires the annotator markings to be *exactly* the same in every aspect; and an 'identification' F_1 that marks

[4] Dr. Finlayson managed the project.

agreement if there is any overlap at all between two annotator's marked instances. Agreements are reported for each text individually, as well as microaveraged over the whole corpus. As would be expected, the strict F_1 result is low (0.27), as any disagreement at all (including disagreement over the exact extent of a function, which might include several hundred tokens) gives a penalty. However, the identification measure measures agreement by performing a "best effort" alignment, where function markings that agree exactly are paired off, and then function markings are then paired off in descending order of degree of overlap. This measure of agreement is very good (0.76), especially considering the complexity of Propp's system.

We also report the Fleiss kappa for identification of the primary function symbols. Although annotators were given the list of functions to mark on the texts, they were allowed to change the symbol identity or subtype if they felt necessary. However, this was not necessary in most cases, as the microaveraged kappa was a "near perfect" 0.92.

For moves, we report a strict and a 'grouping' F_1 measure, where the latter measures an F_1 measure calculated between the function instances involved in moves (rather than the moves as a whole, as for the strict measure). Because these tales were single move, these measures assess only how well annotators marked functions as either "preparatory" or "normal"; as described in our annotation scheme (§3.2), preparatory functions are contained within their own move. 'Grouping' F_1 agreement was good at 0.67.

For dramatis personae, we report a strict and a identification F_1 measures, as well as Fleiss kappa for assignment of dramatis personae labels. The identification F_1 measure marks whether the annotators agreed on whether a character was a dramatis personae, not necessarily on the label(s). Both cumulative F_1 measures are quite high, at 0.82 and 0.88 for the strict and identification, respectively. The cumulative Fleiss Kappa agreement was good at 0.63.

Given these high measures, especially the cumulative F_1 measures, it is evident that Propp's morphology can reliably be applied to narrative texts by human annotators.

6 Related Work

One of the most notable attempts to annotate Proppian morphologies is the Proppian Fairy Tale Markup Language (PftML) [35]. Malec discusses the method and difficulties encountered while creating his annotation scheme and applying it parts of 20 Russian magic tales, but the work is hard to assess as it does not include examples of annotation nor a description of the annotation scheme in the version available online [35]. PftML, as described, does not appear to handle the annotation of implicit functions nor the annotation of dramatis personae and Proppian moves. Further work on PftML brings in additional linguistic information [30, 7] and includes brief examples of annotation, but does not appear to support signals nor discontinuous regions representing functions. Recent work on PftML looks towards the possibility of automatically classifying and annotating Russian folktales [34].

Work in story generation has resulted in ProppOnto, an OWL ontology based on Propp's morphology [38]. Peinado *et al.* explicitly state that their system is not intended to be complete (p. 6) and appear to use the system solely for story generation rather than annotation.

Recent work by Lendvai *et al.* [29, 30] attempts to integrate PftML and ProppOnto together with linguistic information, demonstrating an approach to enrich both schemes. Currently, this work appears to be a proposal, with it being unclear whether or not work has proceeded in this integration. Further, this integration does not alleviate some of the issues that PftML and ProppOnto suffer from as annotation schemes.

Work by Bod *et al.* [4] is the work most suited to direct comparison with our results: they directly studied the reproducibility of Proppian narrative annotations. Bod's study was split into two trials. The first study consisted of nine students who were briefed for 45 minutes, given a small handout describing the functions and dramatis personae, and asked to annotated four single-move stories with functions, subfunctions, and dramatis personae. The second study was similar to the first, but omitted subfunctions, had dramatis personae already assigned, and only had six participants [4, pp. 18–20]. Bod *et al.* concluded that the dramatis personae had an important effect on the assignment of functions. Further, they conclude from the first study that dramatis personae cannot be reliably annotated and from the second study that even given the dramatis personae, human annotators cannot reliably annotate some functions (p. 20).

While Bod *et al.* state that their previous suggestion of a large-scale study to determine how reliably humans can apply Proppian morphology to narratives is "not worthwhile" (p. 21), they themselves admit that making Propp's "vague descriptions" (p. 20) understandable to annotators may require more time and training than they were given in the study. Bod *et al.* suggest that a property necessary for a formal framework to be the basis of an automated system is that sufficiently trained human annotators will annotate a narrative in the same way (p. 17). We believe that our study has sufficiently addressed the training shortcoming identified by Bod *et al.*, by giving annotators more than 30 hours training, as well as another 90 hours of time to produce annotations.

6.1 Use of Propp in Story Generation

A substantial amount of work has been built on top of Propp's morphology in the field of story generation. Some of the first work using Propp to generate stories was done by Klein *et al.* [27, 28], creating a system capable of generating folktale text based on Propp's morphology.

Grasbon and Braun [24] describe an implementation of an interactive story-telling system that used the dependencies and sequences inherent in Propp's functions to generate narratives from pre-written scenarios for each function.

Arinbjarnar [3] describes the creation of a murder mystery game engine based on a Bayesian network designed with a morphology similar to Propp's morphology. Also in games, Fairclough and Cunningham's work [10, 11, 12] integrates Proppian characters and character functions into a game as part of a case-based planning and constraint satisfaction system designed to make agents react appropriately to player actions while following a plot.

Thomas [42] describes two methods for generating folktales using Propp's morphology. The first method expands existing analyses of tales (called *schemes* by Thomas) and filling in roles with random character. The second method attempts to generate fully-formed moves by considering the sequential order of Propp's functions and filling in subtypes for the functions in the move.

Early work by Gervás uses ProppOnto [38] as part of a case-based reasoning system to measure the semantic distance between situations and maintain "an independent story structure from the simulated world" [22, p. 4]. Peinado and Gervás [37] also raise questions about the creativity of narratives generated by Proppian morphology, determining that it is possible to produce relatively novel narratives through generation, but admitting that more experiments are needed. Newer work by Gervás describes the partial implementation of a story generation prototype based on Propp's morphology [21] and work by Gervás, León, and Méndez [23] attempts to reconcile existing schemes with Proppian character functions and extend a Propp-based generation system (Propper) to support schema-driven generation.

7 Contributions

Propp's morphology has had a deep impact in narratology, narrative understanding, and computational approaches to narrative, especially in story generation. Propp's work has inspired and influenced a substantial body of work on narrative structure since its translation into English. ProppML, the scheme we present here, is a complete annotation scheme for Proppian morphologies, succeeding previous work on annotation schemes targeting Propp's work. We have produced a pilot corpus of annotated data larger than any previous study.

We have also shown that Propp's morphology can be reliably applied by human annotators to single-move tales with a cumulative agreement for functions of $F_1^i = 0.76$ and $\kappa = 0.92$; for moves, $F_1^g = 0.67$; and for dramatis personae $F_1^s = 0.82$, $F_1^i = 0.88$, and $\kappa = 0.6$. The definitions for how F_1^i, F_1^g, and F_1^s are calculated is given in §5.

Our work provides a powerful tool for the machine learning of Proppian morphologies. Our lab has already used a prior version of the annotation scheme to demonstrate the machine learning of Propp's morphology [16]; our current version improves on this past scheme and paves the way toward even more accurate systems in the future.

Acknowledgments. This research was made possible by an FIU's Presidential Fellowship and FIU's SCIS's Director's Fellowship, both awarded to W. Victor H. Yarlott. This work was also partially supported by National Institutes of Health (NIH) grant number 5R01GM105033-02. We would like to thank our two annotators, David Roberson and Aeron Commins, and our adjudicator, Jacob Stulberg, for their hard work and attention to detail.

References

1 Antti Amatus Aarne. *Verzeichnis der märchentypen.* Suomalainen tiedeakatemia, 1910.
2 Aleksandr Nikolaevich Afanas'ev. *Narodnye Russkie Skazki.* Moscow: Gos. Izd-vo Khudozh Lit-ry., 1957.
3 María Arinbjarnar. *Murder She Programmed: Dynamic Plot Generating Engine for Murder Mystery Games.* Thesis, Reykavik University, 2005. URL: `http://www-users.cs.york.ac.uk/~maria/greinar/BSc.pdf`.
4 Rens Bod, Bernhard Fisseni, Aadil Kurji, and Benedikt Löwe. Objectivity and Reproducibility of Proppian Narrative Annotations. In Mark Alan Finlayson, editor, *Third Workshop on Computational Models of Narrative (CMN)*, pages 17–21, Istanbul, Turkey, 2012. European Language Resources Association (ELRA).
5 Claude Bremond, Jean Verrier, Thomas G. Pavel, and Marylin Randall. Afanasiev and propp. *Style*, pages 177–195, 1984.
6 Benjamin N. Colby. A Partial Grammar of Eskimo Folktales. *American Anthropologist*, 75:645–662, 1973.
7 Thierry Declerck, Kerstin Eckart, Piroska Lendvai, Laurent Romary, and Thomas Zastrow. Towards a standardized linguistic annotation of fairy tales. In *Workshop on Language Resource and Language Technology Standards*, pages 60–63, 2010.
8 Alan Dundes. From etic to emic units in the structural study of folktales. *The Journal of American Folklore*, 75(296):95–105, 1962.
9 Alan Dundes. *The Morphology of North American Indian Folktales.* Folklore Fellows Communications, 1964.
10 Chris R. Fairclough and Pádraig Cunningham. An Interactive Story Engine. In *Proceedings of the 13th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2002)*, pages 171–176, 2002.

11 Chris R. Fairclough and Padraig Cunningham. A Multiplayer Case Based Story Engine. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, pages 41–47. EUROSIS, 2003. URL: `https://www.scss.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-43.pdf`.

12 Chris R. Fairclough and Pádraig Cunningham. AI Structuralist Storytelling In Computer Games. In *Proceedings of the 5th International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*. University of Wolverhampton, 2004.

13 Mark A Finlayson. Propplearner: Deeply annotating a corpus of russian folktales to enable the machine learning of a russian formalist theory. *Digital Scholarship in the Humanities*, page fqv067, 2015.

14 Mark Alan Finlayson. Collecting Semantics in the Wild: The Story Workbench. In Jacob Beal, Paul Bello, Nick Cassimatis, Michael Coen, and Patrick Winston, editors, *Proceedings of the AAAI Fall Symposium on Naturally Inspired Artificial Intelligence (published as Technical Report FS-08-06, Papers from the AAAI Fall Symposium)*, volume 1, pages 46–53, Arlington, VA, 2008. AAAI Press, Menlo Park, CA. URL: `http://www.aaai.org/Papers/Symposia/Fall/2008/FS-08-06/FS08-06-008.pdf`.

15 Mark Alan Finlayson. The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool. In Emmett Tomai, Jonathan P. Rowe, and David K. Elson, editors, *Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT4)*, pages 21–24, Stanford, CA, 2011. AAAI Press, Menlo Park, CA. URL: `http://aaai.org/ocs/index.php/AIIDE/AIIDE11WS/paper/view/4091`.

16 Mark Alan Finlayson. Inferring Propp's Functions from Semantically-Annotated Text. *Journal of American Folklore, Special Issue on Computational Folkloristics*, 129(511):53–57, 2016.

17 Mark Alan Finlayson and Tomaz Erjavec. Overview of Annotation Creation: Processes & Tools. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*. Springer, 2016. `arXiv:arXiv:1602.05753`.

18 J. L. L. B. Fischer. The Sociopsychological Analysis of Folktales. *Current Anthropology*, 4(3):235–295, 1963. URL: `http://www.jstor.org/stable/2739608`, `doi:10.1086/200639`.

19 Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

20 William B. Frakes and Ricardo Baeza-Yates. *Information retrieval: data structures and algorithms*. Prentice Hall PTR, 1992.

21 Pablo Gervás. Propp's Morphology of the Folk Tale as a Grammar for Generation. In Mark A. Finlayson, Bernhard Fisseni, Benedikt Löwe, and Jan Christoph Meister, editors, *Proceedings of the 4th Workshop on Computational Models of Narrative (CMN'13)*, volume 32, pages 106–122. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013. `doi:10.4230/OASIcs.CMN.2013.106`.

22 Pablo Gervás, Belén Daíz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4-5):235–242, 2005. `doi:10.1016/j.knosys.2004.10.011`.

23 Pablo Gervás, Carlos León, and Gonzalo Méndez. Schemas for Narrative Generation Mined from Existing Descriptions of Plot. In *Proceedings of the 6th Workshop on Computational Models of Narrative (CMN'15)*, pages 54–70, 2015. URL: `http://narrative.csail.mit.edu/cmn15/schedule_cmn15.html`.

24 Dieter Grasbon and Norbert Braun. a morphological approach to interactive storytelling. In *Proceedings of cast01, Living in Mixed Realities*, pages 337–340. FhG – Institut Medienkommunikation (IMK), German Federal Ministry of Education and Research, 2001.

URL: http://netzspannung.org/version1/extensions/cast01-proceedings/pdf/by_name/Grasbon.pdf.

25 A. J. Greimas. *Structual Semantics: An Attempt at a Method*. University of Nebraska Press, Lincoln, Nebraska, 1983.

26 Norbert Guterman. *Russian Fairy Tales*. Pantheon Books, 1973.

27 Sheldon Klein. Meta-compiling text grammars as a model for human behavior. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 84–88. Association for Computational Linguistics, 1975.

28 Sheldon Klein, John F. Aeschlimann, Matthew A. Appelbaum, D. F. Blasiger, Elizabeth J. Curtis, Mark Foster, S. D. Kalish, S. J. Kamin, Y. D. Lee, L. A. Price, et al. Modeling Propp and Lévi-Strauss in a metasymbolic simulation system. *Patterns in Oral Literature*, pages 141–222, 1977.

29 Piroska Lendvai, Thierry Declerck, Sándor Darányi, Pablo Gervás, Raquel Hervás, Scott Malec, and Frederico Peinado. Integration of Linguistic Markup into Semantic Models of Folk Narratives: The Fairy Tale Use Case. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, pages 1996–2001. European Language Resources Association (ELRA), 2010.

30 Piroska Lendvai, Thierry Declerck, Sándor Darányi, and Scott Malec. Propp revisited: Integration of linguistic markup into structured content descriptors of tales. In *Proceedings of the Conference for Digital Humanities 2010*, 2010.

31 Claude Lévi-Strauss. L'analyse morphologique des contes russes. *International Journal of Slavic Linguistics and Poetics*, 3:122–149, 1960.

32 Claude Levi-Strauss. Structure and Form: Reflections on a Work by Vladimir Propp. In Vladimir Propp, editor, *Theory and History of Folklore*, chapter 11, pages 167–210. University of Minnesota Press, Minneapolis, MN, 1984.

33 Isabel Machado, Ana Paiva, and Paul Brna. Real characters in virtual stories. In *Virtual Storytelling Using Virtual Reality Technologies for Storytelling*, pages 127–134. Springer, 2001.

34 Scott Malec. Autopropp: Toward the automatic markup, classification, and annotation of russian magic tales. In *Proceedings of the First International AMICUS Workshop on Automated Motif Discovery in Cultural Heritage and Scientific Communication Texts*, pages 112–115, 2010.

35 Scott A. Malec. Proppian structural analysis and xml modeling. *Proc. of Computers, Literature and Philology (CLiP 2001)*, 2001. URL: https://www.researchgate.net/publication/247286265_Proppian_Structural_Analysis_and_XML_Modeling.

36 Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

37 Federico Peinado and Pablo Gervás. Creativity Issues in Plot Generation. In *Working Notes on Workshop on Computational Creativity, at 19th International Joint Conference on Artificial Intelligence (2nd IJWCC'05)*, pages 45–52. Departamento de Ingeniera del Software e Inteligencia Artificial, Universidad Complutense de Madrid, 2005. URL: http://www.fdi.ucm.es/profesor/fpeinado/publications/2005-peinado-creativity.pdf.

38 Federico Peinado, Pablo Gervás, and Belén Díaz-Agudo. A description logic ontology for fairy tale generation. In *Language Resources for Linguistic Creativity Workshop, 4th LREC Conference*, pages 56–61. Citeseer, 2004.

39 Vladimir Propp. *The Morphology of the Folktale (2nd ed.)*. University of Texas Press, Austin, TX, 1968.

40 James Pustejovsky and Amber Stubbs. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. O'Reilly, Sebastopol, CA, 2013.

41 Thomas Rieger and Norbert Braun. Narrative use of sign language by a virtual character for the hearing impaired. *Computer Graphics Forum*, 22(3):651–660, 2003.

42 Craig Michael Thomas. *The Algorithmic Expansion of Stories*. Thesis, Queen's University, 2010. URL: `http://hdl.handle.net/1974/6127`.

43 Cornelis Joost van Rijsbergen. *Information Retrieval*. London: Butterworths, 1979. URL: `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

44 Roman M Vólkov. *Shazka. Rozyskanija po sjužetosloženiju narodnoj skazki.*, volume 1. Ukrainian State Publishing House, 1924.

A ProppML Detailed Specification

In this appendix we give a detailed description of the annotation scheme. Figure 1 shows an excerpt from an annotated file (The Magic Swan Geese, tale #113). Each annotated file is an XML document with the basic format shown in Figure 2

A Story Workbench annotated file represents a stand-off annotated text, with each annotation layer encoded within a `<rep>` tag, standing for *representation*, which collects together a set of annotations that all share an annotation scheme. Each representation has a unique string identifier in reverse-domain-namespace form, e.g., `edu.mit.story.char`, and contains multiple `<desc>` tags, each of which encodes a single annotation. As shown in Figure 3, each annotation has three XML attributes: a unique id number (`id`), an offset (`off`), and a length (`len`). The id number is unique among all the annotations in the document. The offset refers to the starting character in the characters of the text to which the annotation applies; the length attribute refers to the length of the annotation, in characters.

The first annotation layer of a Story Workbench file is always the *Character* layer, and contains the text to be annotated in a description with id zero. All offsets and lengths for annotations in the remainder of the document refer to the character content of this `<desc>` tag (minus the first and last linebreak characters). The format of the character body of each other `<desc>` tag in the document is determined by the BNF associated with its parent representation. For example, in Figure 1, the representation layer corresponding to Propp's functions is begun by `<rep id="edu mit.semantics.rep.function" ver="0.5.0">`. The first annotation in this layer begins with `<desc id="2555" len="73" off="353">`. The text that immediately follows that tag is formatted according to the BNF in §A.1.

All of the annotations that follow have fields that share a common structure relating to marking spans of characters in the text, as shown in Figure 4. The general mechanism for marking text spans is the *segment*, which is a sequence of consecutive tokens, with none skipped. Segments may be brought together in a *segment set* to mark discontinuous spans. Each segment is encoded as referring to a particular set of token annotation ids.

SegmentSet. A segment set contains zero, one, or more segments. Multiple segments in a set are encoded as separated with commas.

Segment. Segment describe the set of contiguous tokens that make up a span of text. Segments must contain at least one token, and each additional token is appended to the segment encoding using a tilde.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT story (rep+)>
<!ELEMENT rep (desc+)>
<!ATTLIST rep id CDATA #REQUIRED>
<!ELEMENT desc (#CDATA)>
<!ATTLIST desc id ID #REQUIRED>
<!-- both len and off should be a non-negative integer -->
<!ATTLIST desc len CDATA #REQUIRED>
<!ATTLIST desc off CDATA #REQUIRED>
```

Figure 2 DTD that represents the high-level structure of a XML-formatted Story Workbench annotation file.

```
<desc id="#" len="#" off="#">Annotation</desc>
```

Figure 3 Structure of a `desc` XML tag.

```
SegmentSet        ::= "" | Segment | Segment "," SegmentSet
Segment           ::= TokenID | TokenID "~" Segment
TokenID           ::= <postive integer>
```

Figure 4 BNF for SegmentSets.

TokenID. Tokens are one of the lower levels of linguistic annotation that is performed automatically by Story Workbench. TokenIDs refer to the automatically generated id that is a part of the annotation for each token.

Note that in this BNF, the empty string (`""`) indicates that a segment set may be empty. This convention is followed in the other BNFs below.

A.1 BNF for Propp's Functions

Figure 5 shows the BNF for the annotation of functions. A function annotation is split at it's top level into a function tag and one or more instances.

FunctionTag. The function tag contains all the essential information about the function we're annotating: the function type, the function itself, the modifiers, and whether or not the function is inverted.

FunctionType. The three types of functions are defined as follows: `NORMAL` functions are those that are represented by roman letters in Propp's work, as well as $\uparrow$ and $\downarrow$. The `INITIAL` situation is the function α. Finally, `PREPARATORY` functions are those listed with other Greek letters. Within a narrative, there may only be a single `INITIAL` situation, which may not be inverted, has only one instance, has no signal, and has no inversion signal.

Symbol. Function symbols are listed in Propp's Morphology of the Folktale, and can be represented by any alphanumeric string containing the symbols `[a-zA-Z0-9]`. Those function symbols that do not have a corresponding symbol in the alphanumeric character set are written out (e.g. `alpha`, `beta`, `up`, `down`).

```
Annotation       ::= FunctionTag Instances
FunctionTag      ::= FunctionType ":" PreSuperscript ":" Symbol
                     ":" Inverted? ":" Subscript ":" Superscript
FunctionType     ::= "NORMAL" | "PREPARATORY" | "INITIAL"
Symbol           ::= <string>
PreSuperscript   ::= <string>
Subscript        ::= <string>
Superscript      ::= <string>
Inverted?        ::= "true" | "false"
Instances        ::= FuncInstance | FuncInstance Instances
FunctionInstance ::= InstanceType ":" Signal ":" InversionSignal ":" Extent
InstanceType     ::= "ACTUAL" | "ANTECEDANT" | "SUBSEQUENT"
Signal           ::= SegmentSet
InversionSignal  ::= SegmentSet
Extent           ::= SegmentSet
SegmentSet       ::= "" | Segment | Segment "," SegmentSet
Segment          ::= TokenID | TokenID "~" Segment
TokenID          ::= <postive integer>
```

Figure 5 BNF for Propp's functions. Unlisted non-terminals are explained below. <string> means an alphanumeric string, possibly empty.

Superscript, Subscript, Presuperscript. Function subtype modifiers are also listed in Propp's monograph, and can be represented by any alphanumeric string containing the symbols `[a-zA-Z0-9]`. Those modifiers symbols that do not have a corresponding key in the alphanumeric character set may written out.

Inverted? Functions may be marked as "inverted," meaning the function is fulfilled by something semantically opposite of its usual filler, but this is not the same as negation. For example, refusing to wed would be an inversion of W, but not getting caught is not an inversion of Rs.

Instances. Functions have one or more function instances.

FunctionInstance. Each instance of function has four field: (1) the type, (2) the signal, (3) the inversion signal, and (4) the extent.

InstanceType. Most function instances are explicit, in which case the instance type is "ACTUAL". However, for implicit instances, the closest logically related event is marked as the extent of the instance. If the event happens directly before the function, the instance type is "ANTECEDENT", and if it occurs afterwards, the instance type is "SUBSEQUENT". If there is a tie between events, "ANTECEDENT" is preferred.

Signal. Instance signals are the single word or phrase that most strongly indicates the presence of the function, usually a verb.

InversionSignal. Inversion signals are words that signal the inversion of a function, such as "refused" or "not," depending on the context.

```
Annotation       ::= MoveNum "|" Instances
MoveNum          ::= <non-negative integer>
Instances        ::= FunctionInstance | FunctionInstance "|" Instances
FunctionInstance ::= InstanceIndex "," FunctionID
InstanceIndex    ::= <non-negative integer>
FunctionID       ::= <positive integer>
```

Figure 6 BNF for Propp's moves.

Extent. The extent of an instance should be the smallest region that covers the function. The extent of an instance must cover, at the very least, all the signal words and the inversion signal, if there is one. Extents can be continuous or discontinuous. If a verb is marked as a signal, the core arguments to that verb should be included, but non-core arguments should be excluded. Sentence-ending punctuation should only be included when the extent includes the whole sentence or covers words on either side of the sentence boundary.

A.2 BNF for Propp's Moves

Figure 6 shows the BNF for our annotation of moves within a folktale. A move is that part of a tale characterized by starting with a lack or an act of villainy and ending with the resolution of the lack or the defeat of the villain. A move comprises a move number and one or more function instances.

MoveNum. Each move is given a non-negative integer. By convention, move 0 contains the preparatory functions, including the initial function.

Instances. Each move may contain one more function instances. As noted in the body of the paper, functions may have multiple instances that are part of different moves.

FunctionInstance. This refers to the specific instance of a function that is part of a move.

InstanceIndex. Each function instance is represented by two numbers: the id of the funciton annotation of which it is a part, and the index of that function instance in the list of instances for that function. Function instances are indexed starting at zero.

FunctionID. Functions are covered in Section A.1. The function id refers to the automatically generated id for each function annotation. There can be multiple functions in a single move, but there must always be at least one function in a move.

A.3 BNF for Propp's Dramatis Personae

Figure 7 shows the BNF for dramatis personae. Dramatis personae annotations comprise a label and a coreference bundle id.

DramatisPersonae. A string corresponding to one of the seven character archetypes defined by Propp.

```
Annotation       ::= DramatisPersonae "|" CorefID
DramatisPersonae ::= "HERO" | "VILLAIN" | "DONOR" | "HELPER" | "PRINCESS" |
                     "DISPATCHER" | "FALSE HERO"
CorefID          ::= <positive integer>
```

Figure 7 BNF for Propp's dramatis personae.

CorefID. The coreference bundle id refers to the automatically generated id for each coreference bundle annotation.