# Explanations Generation For Web Service Workflow

**Van Duc Nguyen**
Computer Science Department
New Mexico State University, USA
vnguyen@cs.nmsu.edu

**Son Cao Tran**
Computer Science Department
New Mexico State University, USA
tson@cs.nmsu.edu

**Enrico Pontelli**
Computer Science Department
New Mexico State University, USA
epontell@cs.nmsu.edu

## Abstract

The motivation for the work is the challenge of providing textual explanations of automatically generated scientific workflows (e.g., paragraphs that scientists can include in their publications). The extended abstract presents a system which generates explanations for a web service workflow from sets of atoms derived from a collection of ontologies. The system, called `nlgPhylogeny`, demonstrates the feasibility of the task in the *Phylotastic* project, that aims at providing evolutionary biologists with a platform for automatic generation of phylogenetic trees given some suitable inputs.

**2012 ACM Subject Classification** Computing methodologies → Logic programming and answer set programming, Information systems → Web services, Computing methodologies → Natural language generation

**Keywords and phrases** Phylotastic, Grammatical Framework

**Digital Object Identifier** 10.4230/OASIcs.ICLP.2018.14

## 1 Introduction

The *Phylotastic*[1] project is an attempt for making use of phylogeny trees in education or for researching in biology. To perform a phylogeny tree extraction, the project involves in a series of tasks which is invisible to users. From the need of verification that the phylogeny tree is correctly extracted, some method of describing how the phylogeny tree is delivered to user are provided. One popular way is to describe the progress by natural language. The problem of generating natural language explanations has been explored in several research efforts. For example, the problem has been studied in the context of question-answering systems[2], providing recommendations[3], etc.

In this paper, we describe a system called `nlgPhylogeny`for generating natural language explanations for *Phylotastic* project. The system is powered by Grammatical Framework.

[1] `http://phylotastic.org`
[2] `http://coherentknowledge.com`
[3] `http://gem.med.yale.edu/ergo/default.htm`


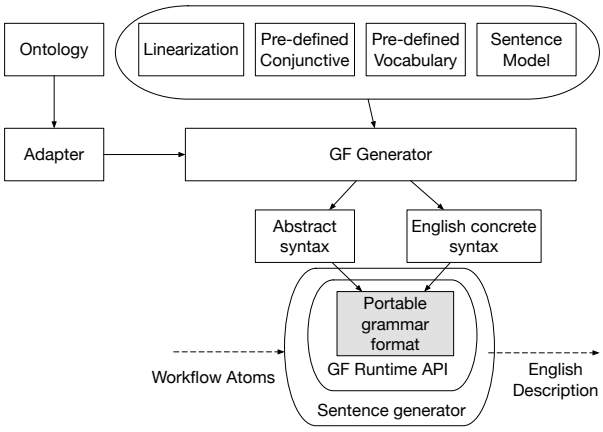


**Figure 1** Overview of `nlgPhylogeny`.

## 2 Methodology

In this section, we describe the `nlgPhylogeny` system. Figure 1 shows the overall architecture of `nlgPhylogeny`. The main component of the system is the *GF generator* whose inputs are the ontology and the elements necessary for the NLG task (i.e., the set of linearizations, the set of pre-define conjunctives, the set of vocabularies, and the set of sentence models) and whose output is a GF program, i.e., a pair of GF abstract and concrete syntax. This GF program is used for generating the descriptions of workflows via the GF runtime API. The adapter provides the GF generator with the information from the ontology, such as the classes, instances, and relations.

### 2.1 Web Service Ontology (WSO)

Phylotastic uses web service composition to generate workflows for the extraction/construction of phylogenetic trees. It makes use of two ontologies: WSO and PO. WSO encodes information about registered web services and their abstract classes. In the following discussion, we refer to a simplified version of the ASP encoding of the ontologies used in [2], to facilitate readability. In WSO, a service has a name and is associated with a list of inputs and a list of outputs.

### 2.2 GF generator

Each Phylotastic workflow is an acyclic directed graph, where the nodes are web services, each consumes some resources (inputs) and produces some resources (outputs). The GF generator produces a portable grammar format (**pgf**) file [1]. This file is able to encode and generate sentences by using GF Runtime API. The GF generator (see Figure 1) accepts two flows of input data:

- The flow of data from the ontology which is maintained by an adapter. The *adapter* is the glue code that connects the ontology to the GF generator. Its main function is to extract classes and properties from the ontology.
- The flow of data from predefined resources that cannot be automatically obtained from the ontology – instead they require manual effort from both ontology experts and linguistic developers;

- A list of *linearizations*; these are essentially the translations of names of ontology entities into linguistic terms. This translation is performed by experts who have knowledge of the ontology domain. An important reason for the existence of this component is that some classes or terms used in the ontology might not be directly understandable by the end user. This may be the result of very specialized strings used in the encoding of the ontology by the ontology engineer (e.g., abbreviations), or the use of URIs for the representation of certain concepts.
- Some *model sentences* which are principally Grammatical Framework syntax trees with meta-information. The meta-information denotes which part of syntax tree can be replaced by some *vocabulary* or *linearization*.
- A list of *pre-defined vocabularies* which are domain-specific for the ontology. A *pre-defined vocabulary* is different from linearizations, in the sense that some lexicon may not be present in the ontology but might be needed in the sentence construction; the predefined vocabulary is also useful to bring variety in word choices when parts of a *model sentence* are replaced by the GF generator.
- A configuration of *pre-defined conjunctives* which depend on the document planning result. Basically, this configuration defines which sentences accept a conjunctive adverb in order to provide generated text transition and smoothness.

Based on the number of inputs and outputs of a service, the GF generator determines how many parameters will be included in the GF abstraction function corresponding to the service. Furthermore, for each input or output of a service, the GF generator includes an *Input* or *Output* in the GF abstract function.

Next, the GF generator looks up in the *sentence models* a model syntax tree whose structure is suitable for the number of inputs and outputs of the service. If such syntax tree exists, the GF generator will replace parts of the syntax tree with the GF service input and output functions, to create a new GF syntax tree which can be appended in the GF concrete function.

From the abstract and concrete syntax built by GF generator, it is possible to generate the sentence

> *The input of service phylotastic_FindScientificNamesFromWeb_GET is a web link and its outputs are a set of species names and a set of scientific names.*

for the atom `occur_concrete(phylotastic_FindScientificNamesFromWeb_GET,1)`. We use the same technique to encode the other types of sentences to describe a full workflow.

## References


1 Krasimir Angelov, Björn Bringert, and Aarne Ranta. PGF: A Portable Run-time Format for Type-theoretical Grammars. *Journal of Logic, Language and Information*, 19:201–228, 2010.

2 Thanh H. Nguyen, Tran Cao Son, and Enrico Pontelli. Automatic Web Services Composition for Phylotastic. In *Practical Aspects of Declarative Languages - 20th International Symposium*, pages 186–202, 2018. doi:10.1007/978-3-319-73305-0_13.