Seed Patterns for Modeling Trees

Aaron Eberhart[1], David Carral[2], Pascal Hitzler[1], Hilmar Lapp[3], and Sebastian Rudolph[2]

[1] Data Semantics (DaSe) Laboratory, Kansas State University, USA
[2] TU Dresden, Germany
[3] Center for Genomic and Computational Biology, Duke University, Durham, NC, USA

Abstract. Trees – i.e., the type of data structure known under this name – are central to many aspects of knowledge organization. We investigate some central design choices concerning the ontological modeling of such trees. In particular, we consider the limits of what is expressible in the Web Ontology Language and provide a reusable ontology design pattern for trees.

1 Introduction

Trees are fundamental data structures for knowledge organization. They make their appearance in the form of taxonomies, meronomies, decision trees, branching processes, etc. As such they are fundamental for ontological knowledge representation.

At the same time, however, it is not possible to fully characterize trees in the Web Ontology Language (OWL) [14,15] (see Section 3). It is thus an important research question how to represent trees in ontology modeling, and to understand the pros and cons of different ways to do it.

We need to realize, of course, that trees in ontology modeling often serve a different purpose than in programming. Operations on trees important in programming include, for example, adding or deleting items or pruning of whole sections; i.e., some of the important operations do actually change the tree. For ontology modeling purposes, in contrast, it is more appropriate to think of a tree as static and as something that is being queried. Typical queries would be to identify roots or leaves, common ancestors, or descendants.

Because trees are an important conceptual tool for knowledge organization, we present a design pattern for modeling trees. We also discuss different design choices related to modeling trees, as well as their respective advantages and disadvantages.

The rest of the paper is structured as follows. In Section 2 we present a particularly interesting use case which has informed our work, namely the use of ontology modeling for evolutionary or phylogenetic trees. In Section 3 we discuss the fundamental shortcomings of the Web Ontology Language (OWL) regarding the modeling of trees. In Section 4 we present a basic ontology design pattern

for the modeling of trees.[4] In Section 5 we discuss the special case of n-bounded trees (e.g., with $n = 2$ for binary trees). Section 6 presents a model for trees that contain data arranged in an order, as well as some general discussion on the representation orders in OWL. In Section 7 we conclude.

This paper is an extended version of [3].

2 Phylogenetic Trees

One of the central tenets following from the theory of organismal evolution is that all life is related through descent with modification [5]. That is, populations of a biological species can over time diverge enough, due to natural selection, adaptation, genetic drift, and other forces acting differentially on different populations, that they form new species, some of which persist and go on themselves to split, giving rise to new species, and so forth. Speciation through diversification can sometimes be driven by new ecologic opportunities, for example when new habitats are being colonized, a process often referred to as adaptive radiation [30,31]. One of the most prominent research objectives in evolutionary science is to reconstruct, using genetic and organismal trait data, the evolutionary history of different organisms, species, or life forms; i.e., to reconstruct the lines of shared descent by which organisms are connected [10,33]. Such a reconstruction is represented in the form of a phylogenetic tree, in which the leaves are often called operational taxonomic units (OTUs) and represent the sampled entities, and internal nodes represent ancestral entities, such as ancestral populations from which descendent ones diverged. Phylogenetic reconstruction results in unrooted trees; the root is normally not known (and cannot normally be sampled), but reasonably accurate mechanisms for inducing a root exist [16,20] (for example, by including in the reconstruction analysis a group of species – a so-called "outgroup" – that are already known to fall outside of the ingroup for which evolutionary patterns are being studied).

A phylogenetic tree represents important evolutionary hypotheses about shared history. For example, two OTUs A and B are more closely related to each other than to OTU C if A and B share a more recent common ancestor than they do with C. The subtree descending from a node forms a clade, clades which share a parent are called sister clades. One of the major objects of comparative phylogenetics is to identify the properties and processes (organismal traits, geographic range, tempo and mode of evolution, etc) by which one clade differs from others, in particular its sisters, and how these properties change along lines of descent in the tree [9,24]. This gives rise to a number of important queries when mapping data onto phylogenetic trees for (or as a result of) analysis. Particularly ubiquitous operations on trees include the following: (1) finding the most recent common ancestor of a given number of nodes (usually leaf nodes); (2) enumerating the leaf nodes, or all nodes descending from a given (internal) node; (3) enumerating the sequence of ancestors of a node to the root;

[4] The pattern is available from `http://ontologydesignpatterns.org/wiki/Submissions:Tree_Pattern`

and (4) identifying the last ancestor of a node A from which another node B is not also descended. We will come back to these and other operations as part of the competency questions for our modeling in Section 4.

Operations (1) and (4) correspond to two principle ways in which the semantics of clade concepts can be defined on a tree [25], whether using a concrete instantiation of a tree, or a hypothetical one. In the field of phylogenetic taxonomy [26], a clade concept defined by the most recent common ancestor of a set of (usually leaf) nodes includes the common ancestor and is referred to as a node-based definition. In contrast, a branch-based definition circumscribes the clade as the last ancestor of a (usually leaf) node that excludes (i.e., does not have as a descendant) another node (also usually a leaf node). The semantics of a clade concept defined in this way is such that the branch subtending from the ancestor node to its parent is included (hence the name "branch-based"). To understand this, remember that a phylogenetic tree is a model of evolutionary lines of descent reconstructed from sampled data. In reality, there may be lines of descent which were not observed (sampled), for example because all organisms from those lines are now extinct, but which, had they been observed, would originate from the subtending branch and which would therefore still be included in the clade because they would branch off after the lineage to be excluded.

It is worth noting that spurred in part by the exponentially increasing amount of data available for phylogenetic reconstruction, very large trees encompassing up to tens of thousands of taxa have recently become available [11,7,8,32,17], culminating in the initial publication of the synthesized Open Tree of Life with about 2 million tips [12]. Such encompassing trees open up unprecedented opportunities for comparative phylogenetic research. However, this also means that our knowledge about the evolution of life is changing at increasing pace and breadth, which makes it necessary to efficiently map clade definitions from one tree to another, or from one revision of the Open Tree of Life to a future one. A recent initiative, termed "phyloreferencing" (http://phyloref.org) aims to accomplish this by using machine reasoning over ontological representations of the semantics of both clade definitions and phylogenetic trees [4,22,28]. In the rest of this paper, we abstract from the specific use case and look at the task of ontological modeling of trees in general.

3 Fundamental Limitations Regarding Tree Modeling

In order to investigate to what degree tree-based properties can be expressed using common knowledge representation (KR) formalisms, we first need to formally define what structures we denote by the notion "tree".

Definition 1. *A* rooted directed branching tree *(short:* tree*) is defined as a directed graph* $T = (V, E)$ *where* V *is a set called* vertices *or* nodes *and* $E \subseteq V \times V$ *is the set of edges, satisfying the following properties:*

1. *There is exactly one node* $r \in V$ *called* root*, which has no incoming edges, i.e.,* $E \cap (V \times \{r\}) = \emptyset$.

2. *Every node $v \in V \setminus \{r\}$ that is not the root has exactly one incoming edge, i.e., there exists exactly one $v' \in V$ such that $(v', v) \in E$. We then call v' the* parent *of v and v the* child *of v'.*
3. *Every node $v \in V$ can be reached from the root traversing edges, i.e., there is a number $n \geq 0$ and a sequence $(v)_{i \in \{0,\dots,n\}}$ such that $r = v_0$, $v = v_n$, and for all $i \in \{0, \dots, n-1\}$ we have $(v_i, v_{i+1}) \in E$.*

A node without children is called leaf node. *A* binary tree *is a tree where every node that is not a leaf has at most two children. An n-*ary tree *is a tree where every node that is not a leaf has exactly n children. An n-*bounded tree *is a tree where every node has at most n children. A tree is* finite *if V is finite.*

When modeling trees using some logic-based KR language, we would like create a knowledge base which has exactly all (finite) trees as its models (possibly using additional auxiliary vocabulary). Unfortunately, it is not too hard to show that this is not possible by any KR formalism that is expressible in first order predicate logic (FOL). A very helpful tool for showing this is the well-known compactness theorem of first-order logic [6].

Theorem 1 (Compactness of FOL). *A set Φ of FOL sentences is satisfiable if and only if every finite subset of Φ is.*

We now use this theorem to show our negative result.

Proposition 1. *Let ψ be a FOL sentence (using the binary predicate "edge") such that every finite tree $T = (V, E)$ corresponds to some model $\mathcal{I}$ of ψ, i.e., $(V, E) \cong (\Delta^{\mathcal{I}}, edge^{\mathcal{I}})$. Then, ψ also has a model which does not correspond to any (finite or infinite) tree.*

Proof. Consider the following sequence $(\varphi_i)_{i \in \mathbb{N}}$ of FOL sentences (where a is a fresh constant):

$$\varphi_1 := \exists x_1 . edge(x_1, a)$$
$$\varphi_2 := \exists x_1 \exists x_2 . edge(x_2, x_1) \wedge edge(x_1, a)$$
$$\varphi_3 := \exists x_1 \exists x_2 \exists x_3 . edge(x_3, x_2) \wedge edge(x_2, x_1) \wedge edge(x_1, a)$$
$$\vdots$$

In short, φ_k expresses that the node a has an incoming edge-path of length k. Now let $\Phi := \{\psi\} \cup \{\varphi_k \mid k \in \mathbb{N}\}$. Obviously, every finite subset of Φ is satisfiable (intuitively, just pick an arbitrary large finite tree and then pick a such that it is "deep enough" in the tree). Then, by compactness of FOL, Φ itself must be satisfiable. However, in a model $\mathcal{I}$ of Φ the element $a^{\mathcal{I}}$ cannot be reachable from the root, since then it would have an incoming edge-path of maximal length which cannot be the case by construction of Φ. Hence $\mathcal{I}$ cannot correspond to a tree. By construction, $\mathcal{I}$ is also a model of ψ. □

This result shows, that trees (finite or infinite) are not fully axiomatizable in FOL and any attempt to do so will only be approximate (although potentially useful).

On the other hand, trees are axiomatizable when we extend FOL (or just DLs for that matter) by a transitive closure operator for binary predicates. Assume that, for every binary predicate (or in DL terms: role) p, we allow for a binary predicate/role name p^+ and define its semantics such that $(p^+)^{\mathcal{I}}$ is the transitive closure of $p^{\mathcal{I}}$. Then the conditions of Definition 1 can be expressed using the following axioms:

$$\text{RootNode} \sqsubseteq \{\text{root}\} \tag{1}$$

$$\{\text{root}\} \sqsubseteq \neg\exists \text{edge}^-.\top \tag{2}$$

$$\neg\{\text{root}\} \sqsubseteq =1\,\text{edge}^-.\top \tag{3}$$

$$\neg\{\text{root}\} \sqsubseteq \exists(\text{edge}^-)^+.\{\text{root}\} \tag{4}$$

To axiomatize the class of binary trees, the following axiom can be added:

$$\top \sqsubseteq \neg\exists \text{edge}.\top \sqcup =2\,\text{edge}.\top \tag{5}$$

In order to impose finiteness, one can axiomatize (as an auxiliary additional structure) a finite linear order with a starting element and an ending element and the successor role:

$$\{\text{start}\} \sqsubseteq \neg\exists \text{succ}^-.\top \tag{6}$$

$$\neg\{\text{start}\} \sqsubseteq =1\,\text{succ}^-.\top \tag{7}$$

$$\neg\{\text{start}\} \sqsubseteq \exists(\text{succ}^-)^+.\{\text{start}\} \tag{8}$$

$$\{\text{end}\} \sqsubseteq \neg\exists \text{succ}.\top \tag{9}$$

$$\neg\{\text{end}\} \sqsubseteq =1\,\text{succ}.\top \tag{10}$$

$$\neg\{\text{end}\} \sqsubseteq \exists(\text{succ})^+.\{\text{end}\} \tag{11}$$

Transitive closures are not part of the OWL specification [14], so this characterization cannot be used when modeling in the Web Ontology Language. Description logics featuring regular expressions over roles have, however, been considered since the early days of DL research [1] and decision and query answering procedures have been described for very expressive DLs with that feature [2].

4 A Simple Tree Pattern

The repository of ontology design patterns on ontologydesignpatterns.org does not contain any pattern for trees. There is also none for graphs that could have been used to specialize a trees pattern. The repository contains a pattern for lists, though,[5] and a list pattern could be generalizable to a tree pattern.

The schema diagram for this list pattern is depicted in Figure 1. It reuses the sequence pattern[6] which seems to be the relevant part for our purposes. We depict the sequence pattern schema diagram in Figure 2 and all non-tautological axioms are given in Figure 3.[7] The axiomatization appears to be rather minimalistic, e.g., "follows" should be transitive over "directlyFollows", and for a sequence we should also use cardinality restrictions to limit the number of followers and predecessors. We return to this later on.

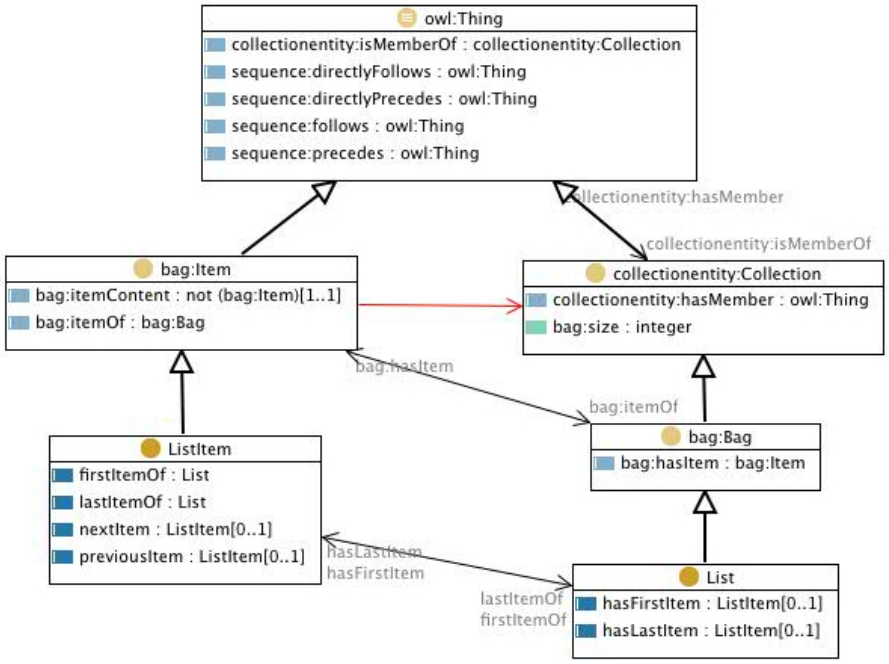


Fig. 1. List pattern schema diagram from ontologydesignpatterns.org

The list pattern just cited provides basic building blocks for a simple tree pattern. However, we opt to change the names of the properties: It seems to be more appropriate to use "hasChild" and "hasDescendant" rather than "directlyPrecedes" and "precedes", and to use "hasParent" and "hasAncestor" rather than "directlyFollows" and "follows."

Before proceeding with the tree pattern, we present a set of competency questions [18] which seem representative to us and include operationes raised as important in Section 2:

1. Determine the root.
2. Determine all ancestors of a given node.
3. Determine all leaves.
4. Determine all descendants of a given node.
5. Determine all descendants of a given node which are leaves.
6. Given two nodes, determine whether one is a descendant of the other.
7. Given two nodes, determine all commmon ancestors.

[5] `http://ontologydesignpatterns.org/wiki/Submissions:List`
[6] `http://ontologydesignpatterns.org/wiki/Submissions:Sequence`
[7] Generated with the OWLAPI LaTeX renderer [29].

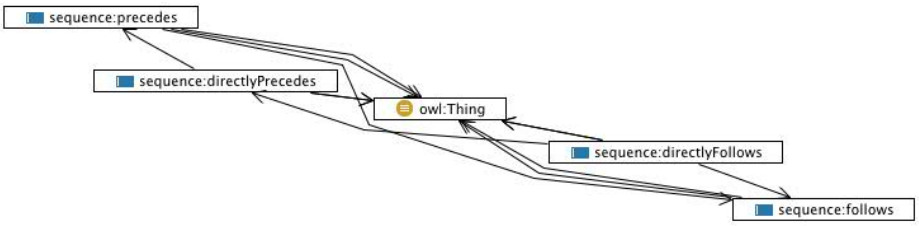


Fig. 2. Sequence pattern schema diagram from ontologydesignpatterns.org

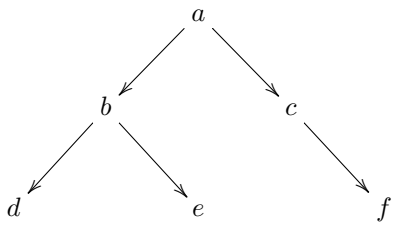
$$\begin{aligned}
\text{directlyFollows} &\sqsubseteq \text{follows} \\
\text{directlyFollows} &\equiv \text{directlyPrecedes}^- \\
\text{directlyPrecedes} &\sqsubseteq \text{precedes} \\
\text{precedes} &\equiv \text{follows}^- \\
&\text{TransitiveProperty(follows)} \\
&\text{TransitiveProperty(precedes)}
\end{aligned}$$

Fig. 3. Axioms for the sequence pattern from Figure 2. We omitted axioms that were tautologies.

8. Given two nodes, determine the latest common ancestor. This means the node that is an ancestor of the two nodes but has no children that are themselves ancestors of both nodes.
9. Given two nodes x and y, determine the earliest ancestor of x which is not an ancestor of y.

We next give our proposal for a simple tree pattern. Afterwards we will discuss our design choices. The schema diagram is given in Figure 4. However, the axiomatization is really much more important; it can be found in Figure 6. Note that some additional desired axioms, such as hasParent $\sqsubseteq$ hasAncestor and transitivity of hasAncestor can be inferred from the ones stated.

Before we proceed, let us first make a concrete example how this pattern informs the graph structure of the ABox. Given a tree such as



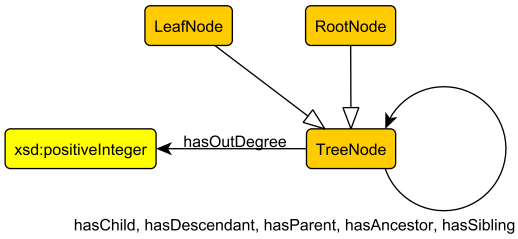


Fig. 4. Schema diagram for the simple tree pattern. Unlabelled arrows are subclass relationships.

RootNode(a)	LeafNode(d)	LeafNode(e)
LeafNode(f)	TreeNode(b)	TreeNode(c)
hasChild(a, b)	hasChild(a, c)	hasChild(b, d)
hasChild(b, e)	hasChild(c, f)	hasSibling(b, c)
hasSibling(d, e)	hasOutDegree($a, 2$)	hasOutDegree($b, 2$)
hasOutDegree($c, 1$)	hasOutDegree($d, 0$)	hasOutDegree($e, 0$)
hasOutDegree($f, 0$)		

Fig. 5. Example ABox for a tree.

we encode it using the ABox from Figure 5; note that we omit redundant statements which can be inferred from the axioms.

The axioms from Figure 6 should be self-explanatory. Axiom (17) uses a datatype facet. The complete set of axioms is not in OWL DL because axioms (33) and (34) declare irreflexivity for non-simple roles. If it is desireable to stay within OWL DL, these axioms could be omitted.

Our pattern and axiomatization include some terms which may appear to be redundant. Indeed, we could have omitted the use of hasParent and hasAncestor as these are simply inverses of hasChild and hasDescendant, respectively. Other aspects, however, are not redundant.

The property hasOutDegree may appear to be redundant, as it captures the number of children of a node. However, it is not redundant as far as the OWL model is concerned, because the underlying open world assumption makes it impossible to count the number of children. This could of course be addressed using a (local) closed world extension of OWL, such as in [21], however the current standard does not support this. For the same reason, membership in the classes RootNode and LeafNode cannot be inferred.

The case of the hasSibling property is more intricate. Again it would naively appear as if it were redundant. However, we have not been able yet to axiomatize

$$\text{LeafNode} \sqsubseteq \text{TreeNode} \tag{12}$$
$$\text{RootNode} \sqsubseteq \text{TreeNode} \tag{13}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasOutDegree.xsd:positiveInteger} \tag{14}$$
$$\text{TreeNode} \sqsubseteq =1\text{hasOutDegree.xsd:positiveInteger} \tag{15}$$
$$\text{LeafNode} \equiv \text{TreeNode} \sqcap \forall \text{hasOutDegree.}\{0^{\wedge\wedge}\text{xsd:positiveInteger}\} \tag{16}$$
$$\text{TreeNode} \sqcap \neg\text{LeafNode} \equiv \text{TreeNode} \sqcap \forall \text{hasOutDegree.}\{x^{\wedge\wedge}\text{xsd:positiveInteger} \mid 1 \leq x\} \tag{17}$$
$$\text{hasChild} \equiv \text{hasParent}^{-} \tag{18}$$
$$\text{hasDescendant} \equiv \text{hasAncestor}^{-} \tag{19}$$
$$\text{hasChild} \sqsubseteq \text{hasDescendant} \tag{20}$$
$$\text{hasDescendant} \circ \text{hasDescendant} \sqsubseteq \text{hasDescendant} \tag{21}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasChild.TreeNode} \tag{22}$$
$$\text{TreeNode} \sqcap \neg\text{LeafNode} \equiv \text{TreeNode} \sqcap \exists \text{hasChild.TreeNode} \tag{23}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasDescendant.TreeNode} \tag{24}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasParent.TreeNode} \tag{25}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasSibling.TreeNode} \tag{26}$$
$$\text{TreeNode} \sqcap \neg\text{RootNode} \equiv \text{TreeNode} \sqcap =1\text{hasParent.}\top \tag{27}$$
$$\text{TreeNode} \sqsubseteq \forall \text{hasAncestor.TreeNode} \tag{28}$$
$$\text{RootNode} \equiv \text{TreeNode} \sqcap \neg\exists \text{hasParent.}\top \tag{29}$$
$$\text{LeafNode} \equiv \text{TreeNode} \sqcap \neg\exists \text{hasChild.}\top \tag{30}$$
$$\text{Irreflexive(hasChild)} \tag{31}$$
$$\text{Irreflexive(hasParent)} \tag{32}$$
$$\text{Irreflexive(hasDescendant)} \tag{33}$$
$$\text{Irreflexive(hasAncestor)} \tag{34}$$
$$\text{hasSibling} \equiv \text{hasSibling}^{-} \tag{35}$$
$$\text{Irreflexive(hasSibling)} \tag{36}$$

Fig. 6. Axioms for the tree pattern from Figure 4.

it in the general case; for special cases where it is possible see Section 5. A naive attempt by means of a rule[8]

$$\text{hasParent}(x, y) \wedge \text{hasChild}(y, z) \rightarrow \text{hasSibling}(x, z)$$

is insufficient because for addressing some competency questions we require irreflexivity of hasSibling, while the rule above renders hasSibling to be non-simple, which is not allowed together with irreflexivity in OWL DL.

Let us return to the competency questions listed earlier; it turns out we can address them all even when omitting the irreflexivity axioms for hasDescendant and hasAncestor, such that our model stays within OWL DL. Each answer takes the form of a query that can be formalized using DL concepts containing the elements we are interested in. Questions 1 and 3 can be addressed using the RootNode and LeafNode classes. Questions 2 and 4 are straightforward, as is question 5 using the LeafNode class. Question 6 can be solved with two queries using the hasDescendant property.

The remaining questions are more intricate. Given two nodes x and y, Question 7 can be addressed via

$$\exists \text{hasDescendant}.\{x\} \sqcap \exists \text{hasDescendant}.\{y\}.$$

Question 8 seems to require use of the hasSibling property.[9] For readability we first give the solution as first-order predicate logic formula:

$$\begin{aligned}\forall w, x, y, z \quad & \text{hasChild}(z, w) \wedge (w = x \vee \text{hasDescendant}(w, x)) \\ & \wedge \text{hasSibling}(w, v) \wedge (v = y \vee \text{hasDescendant}(v, y)).\end{aligned}$$

Conversion of this into OWL following the approach laid out in [19] results in the class description.

$$\begin{aligned}\exists \text{hasChild}.((\{x\} \sqcup \exists \text{hasDescendant}.\{x\}) \\ \sqcap (\exists \text{hasSibling}.(\{y\} \sqcup \exists \text{hasDescendant}.\{y\})).\end{aligned}$$

In a similar fashion, Question 9 can be addressed using the class description

$$(\{x\} \sqcup \exists \text{hasDescendant}.\{x\}) \sqcap (\exists \text{hasSibling}.(\{y\} \sqcup \exists \text{hasDescendant}.\{y\})).$$

[8] This rule can be converted into OWL DL using rolification, see [19].

[9] A naive attempt such as

$$\begin{aligned}&\text{TreeNode} \sqcap \exists \text{hasDescendant}.\{x\} \sqcap \exists \text{hasDescendant}.\{y\} \\ &\quad \sqcap \exists \text{hasChild}.((\{x\} \sqcup \exists \text{hasDescendant}.\{x\}) \sqcap \neg\{y\} \sqcap \neg\exists \text{hasDescendant}.\{y\}) \\ &\quad \sqcap \exists \text{hasChild}.((\{y\} \sqcup \exists \text{hasDescendant}.\{y\}) \sqcap \neg\{x\} \sqcap \neg\exists \text{hasDescendant}.\{x\})\end{aligned}$$

is insufficient as hasDescendant is just a transitive superrole over hasChild, so non-existence in the desired form cannot be guaranteed. Using a non-monotonic (essentially, second-order) extension of the underlying description logic, such as circumscription, would make it possible, in a way very similar to [21].

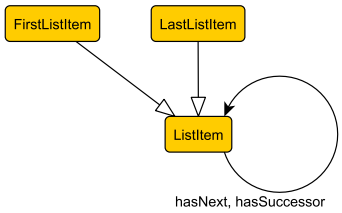


Fig. 7. Schema diagram for the simple list pattern, derived from the tree pattern.

Note that irreflexivity of hasSibling is required for the class descriptions just given. Or to be more precise, what is required is that there is no node z in the tree for which hasSibling(z, z) is declared or can be inferred; note that this is in fact a weaker requirement than what we get by declaring irreflexivity. As a consequence, the irreflexivity declaration for hasSibling can actually be omitted from the axiomatization without impact on the just given solution to Question 9; however we prefer to keep the irreflexivity declaration in the axiomatization as it disambiguates the model [13]. As mentioned earlier, we have not been able to find a solution to infer the (irreflexive) hasSibling relationship in the general case using OWL axioms, thus we require it as a primitive. We will revisit this in the next section, though.

Let us finally use our tree pattern to recover a list pattern based on it. The schema diagram can be found in Figure 7. We keep only one property, hasNext, which corresponds to hasChild, and hasSuccessor, which corresponds to hasDescendant. The root becomes the first list item, the leaves become the last list item. The outdegree is always 1 unless it's the last list item, so we also omit this information. The corresponding axiomatization, as derived from the tree axiomatization above, can be found in Figure 8. As before, Axiom (47) causes the pattern to fall outside OWL DL, and if this is undesirable, this axiom should be omitted.

5 Trees With Bounded Arity

In this section, we look at n-bounded trees as a special case, and present a set of OWL axioms that can be employed to model these.

Definition 2. *A* rooted directed branching n-bounded tree *(short:* n-bounded tree*) is a tree* $T = (V, E)$ *where every node has at most* n *outgoing edges; i.e., for every* $v \in V$*,* $|E \cap (\{v\} \times V)| \leq n$*.*

In the previous section, we have discussed why we needed to include an explicit hasSibling relation in our model, namely because we were unable to axiomatically define it in OWL. If we know that the trees under consideration are n-bounded, though, we can in fact infer the hasSibling relation.

$$\text{FirstListItem} \sqsubseteq \text{ListItem} \tag{37}$$
$$\text{LastListItem} \sqsubseteq \text{ListItem} \tag{38}$$
$$\text{ListItem} \sqsubseteq \forall \text{hasNext}.\text{ListItem} \tag{39}$$
$$\text{ListItem} \sqsubseteq \forall \text{hasNext}^{-}.\text{ListItem} \tag{40}$$
$$\text{ListItem} \sqcap \neg\text{LastListItem} \equiv \text{ListItem} \sqcap =1\text{hasNext}.\text{ListItem} \tag{41}$$
$$\text{ListItem} \sqcap \neg\text{FirstListItem} \equiv \text{ListItem} \sqcap =1\text{hasNext}^{-}.\text{ListItem} \tag{42}$$
$$\text{FirstListItem} \equiv \text{ListItem} \sqcap \neg\exists \text{hasNext}^{-}.\top \tag{43}$$
$$\text{LastListItem} \equiv \text{ListItem} \sqcap \neg\exists \text{hasNext}.\top \tag{44}$$
$$\text{hasNext} \sqsubseteq \text{hasSuccessor} \tag{45}$$
$$\text{hasNext} \circ \text{hasSuccessor} \sqsubseteq \text{hasSuccessor} \tag{46}$$
$$\text{Irreflexive}(\text{hasSuccessor}) \tag{47}$$

Fig. 8. Axioms for the lists pattern from Figure 7.

$$n\text{-BoundedTreeNode} \sqsubseteq \text{TreeNode} \tag{48}$$
$$n\text{-BoundedTreeNode} \sqsubseteq \forall \text{hasAncestor}.n\text{-BoundedTreeNode} \tag{49}$$
$$n\text{-BoundedTreeNode} \sqsubseteq \forall \text{hasDescendant}.n\text{-BoundedTreeNode} \tag{50}$$
$$n\text{-BoundedTreeNode} \sqsubseteq \text{Child}_1 \sqcup \ldots \sqcup \text{Child}_n \tag{51}$$
$$\{\text{Child}_i \sqcap \text{Child}_j \sqsubseteq \bot \mid 1 \leq i < j \leq n\} \tag{52}$$
$$\{n\text{-BoundedTreeNode} \sqsubseteq \leq 1\text{hasChild}.\text{Child}_i \mid 1 \leq i \leq n\} \tag{53}$$
$$n\text{-BoundedTreeNode} \sqsubseteq \leq n\text{hasChild}.n\text{-BoundedTreeNode} \tag{54}$$
$$\{\text{Child}_i \sqsubseteq \exists R_{\text{i}}.\mathsf{Self} \mid 1 \leq i \leq n\} \tag{55}$$
$$\{R_{\text{i}} \circ \text{hasParent} \circ \text{hasChild} \circ R_{\text{j}} \sqsubseteq \text{hasSibling} \mid 1 \leq i < j \leq n\} \tag{56}$$

Fig. 9. Axioms for the n-bounded Tree Pattern to add to the axioms from Figure 6

To this end, we introduce a set of additional axioms (see Figure 9) that, if combined with the axioms from Figure 6 can be used to axiomatize the structure of an n-bounded tree. Key to this are axioms (51) and (52), which together limit the number of children per node to a maximum of n.

Note that, if these axioms are considered, we can indeed automatically infer the hasSibling relationship: Having an upper bound on the number of children per node, we can use a finite set of concept names Child_i to differentiate the children of any given node in the tree. Note how, due to (51) and (52), every n-ary tree node is typed by one and only one class Child_i. Moreover, axioms in (53) enforce that at most one child of every node is in the class Child_i for every $i = 1, \ldots, n$. Using axioms in (55), we automatically infer that each child of every node is connected to itself via some property R_i for some $i = 1, \ldots, n$. Using axiom (56), we can infer the hasSibling relation.

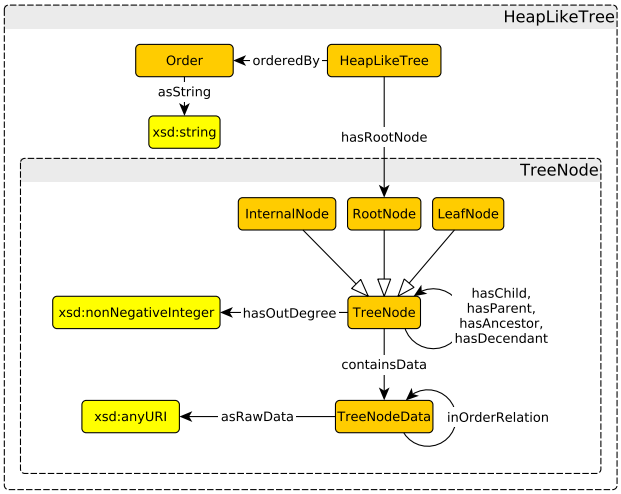


Fig. 10. Schema diagram a heap-like tree, derived from the tree pattern.

Note, though, that hasSibling is non-simple, i.e. the declaration of irreflexivity for hasSibling from Figure 6 violates the role regularity requirements for OWL DL. If the ontology shall fall within OWL DL, the irreflexivity axiom should be removed.

Note also, that the approach just spelled out may not be practical for large n, as the number of models to be checked, e.g. by a tableaux-based reasoner, will increase exponentially with n due to the disjunction in (51).

6 Heap-Like Trees

Another common use case for trees is to arrange tree data in an order based on the child/parent relationship, like in a heap. Even though typically referred to as just a tree, this organizational strategy specifically makes use of an underlying partial order in the data, such as the natural ordering of the integers. Formally, the tree-order of a tree is the partial order $\leq$ of the vertexes such that $v \leq u$ if and only if the unique path from the root to u contains v. Ordering the data in a heap-like tree based on its tree-order with respect to the child relation places useful constraints on how it can occur in the tree, so that, for instance, the root node of a subtree will always be a maximum (or minimum) value occurring within the subtree. A schema diagram for the heap-like tree is shown in Figure 10. In addition to the original axioms in Figure 6, the new semantics are captured by adding the DL axioms in Figure 11 along with DL-Safe rules [23] in Figure 12.

Using DL-Safe rules will restrict us to named individuals, however in the case of axioms about the data in a finite tree this should not much of a loss

$$\text{HeapLikeTree} \sqsubseteq \leqslant 1 \text{hasRootNode.RootNode} \tag{57}$$
$$\text{HeapLikeTree} \sqsubseteq = 1 \text{orderedBy.Order} \tag{58}$$
$$\text{Order} \sqsubseteq \exists \text{asString.xsd:string} \tag{59}$$

$$\text{TreeNode} \sqsubseteq = 1 \text{containsData.TreeNodeData} \tag{60}$$
$$\text{InternalNode} \equiv \text{TreeNode} \sqcap \neg \text{LeafNode} \tag{61}$$

$$\text{TreeNodeData} \sqsubseteq = 1 \text{asRawData.xsd:anyURI} \tag{62}$$
$$\text{TreeNodeData} \sqsubseteq \forall \text{inOrderRelation.TreeNodeData} \tag{63}$$
$$\exists \text{inOrderRelation.TreeNodeData} \sqsubseteq \text{TreeNodeData} \tag{64}$$

$$\text{Asymmetric(inOrderRelation)} \tag{65}$$

Fig. 11. DL axioms to add to the tree pattern for heap-like trees

$$\text{hasDescendant}(x, y) \wedge \text{containsData}(x, a) \wedge \text{containsData}(y, b) \rightarrow \text{inOrderRelation}(a, b) \tag{66}$$
$$\text{hasAncestor}(x, y) \wedge \text{containsData}(x, a) \wedge \text{containsData}(y, b) \rightarrow \text{inOrderRelation}(b, a) \tag{67}$$

Fig. 12. DL-Safe Rules for the heap-like tree pattern from Figure 11

in many cases: data that is known will be ordered, and unlike nodes, unknown data should not be ordered without a known value. The DL-Safe axioms could also be rewritten with role chains in OWL if necessary, avoiding the named individuals requirement, but this would cause inOrderRelation to be a non-simple role and thus conflict with the asymmetry axiom (65) thus violating global constraints imposed on OWL DL. The minor compromise of DL-Safe axioms seems preferable over an OWL Full ontology.

OWL cannot be used to model heap-like trees using their actual data values, since DataProperties do not relate literals to other literals in the same way that roles relate individuals to one another. SWRL allows literal comparison, but only with a predefined set of built-ins. To remedy this, we have created a class called TreeNodeData that contains individuals we can order using the role inOrderRelation, and each has a link to exactly one datum via asRawData. The asymmetry of the role inOrderRelation may seem counter-intuitive for representing a partial order. However, we must make a distinction between the order of tree nodes, which are all unique, and the order of the data that the nodes contain, which may be duplicated. The order in our heap-like tree pattern corresponds to the order of the nodes by using asymmetry instead of antisymmetry to represent the order of children nodes from the root node. If the descendant of a node contains data equal to its own data, both nodes are nonetheless unique nodes in the tree

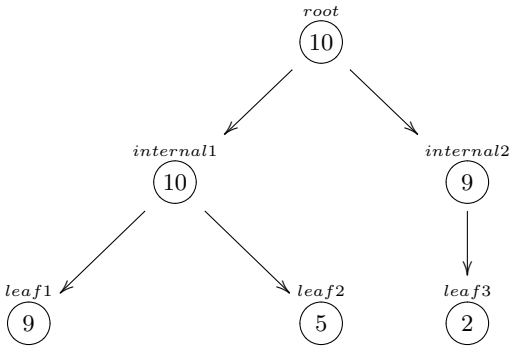


Fig. 13. Example of a max-heap ordered by $\leq$, with labeled nodes

OrderedOutTree(tree)	orderedBy(tree,order)	asString(order,"$\leq$")
hasRoot(tree,root)	containsData(root,dR)	asRawData(dR,**10**)
hasChild(root,internal1)	containsData(internal1,di1)	asRawData(di1,**10**)
hasChild(internal1,leaf1)	containsData(leaf1,dl1)	asRawData(dl1,**9**)
hasChild(internal1,leaf2)	containsData(leaf2,dl2)	asRawData(dl2,**5**)
hasChild(root,internal2)	containsData(internal2,di2)	asRawData(di2,**9**)
hasChild(internal2,leaf3)	containsData(leaf3,dl3)	asRawData(dl3,**2**)

$$\text{hasDescendant(root,leaf2)} \wedge \text{containsData(root,dR)} \wedge \text{containsData(leaf2,dl2)} \rightarrow \text{inOrderRelation(dR,dl2)}$$

$$\text{hasAncestor(leaf3,root)} \wedge \text{containsData(leaf3,dl3)} \wedge \text{containsData(root,dR)} \rightarrow \text{inOrderRelation(dR,dl3)}$$

Fig. 14. Rules made from figures 12 showing order in figure 13

and should be considered unequal. The inOrderRelation role simply provides a way to tie the ordering of the data to the ordering of the nodes. At the end of this section we will discuss antisymmetry and the more general case of strict and non-strict partial orders in OWL.

Examples of trees that can be represented with the heap-like tree pattern are Phylogenetic Trees ordered by common ancestor and Class Hierarchies ordered by $\subseteq$, which already contain data that is ordered in a heap-like way, as well as many trees common in Computer Science, such as, unsurprisingly, a heap ordered by $\leq$ or $\geq$. In Figure 13 we show a max-heap tree ordered by $\leq$, and in Figure 14 we show example facts and rule applications that produce an order.

Regarding the competency questions for trees, the original questions can all be answered in the same way for a heap-like tree as for a general tree. We also add competency questions for heap-like trees. Note that data means members of the TreeNodeData class and not data literals.

Irreflexive(directlyFollows)
Irreflexive(inStrictPartialOrder)

directlyFollows $\sqsubseteq$ inStrictPartialOrder
directlyFollows $\circ$ inStrictPartialOrder $\sqsubseteq$ inStrictPartialOrder

Fig. 15. A strict partial order made from two roles

10. Determine the maximal data element in the tree.
11. For any two tree nodes determine if their data is in an order relation.
12. For any two nodes with data in an order relation determine which is maximal.
13. For any two nodes with data *not* in an order relation, determine what other nodes have data that both nodes' data are in an order relation with.

The answer to Question 10 is simply the data in the root, which is known for any ordered tree. Question 11 can be answered by checking whether the nodes are ancestors or descendants of one another and applying the corresponding DL-safe rule. In Question 12, the maximal data will be the data in the node that is the ancestor of the other node: it has the data that is closest in the order relation to the root. Question 13 can be answered by answering Question 7, except in this case we also know that a common ancestor will have data that corresponds to solutions for DL-Safe rules for both of the nodes and their data.

Although the pattern we have shown is sufficient for modeling heap-like trees, it nonetheless raises interesting questions about the general use of orders in OWL. For instance, if we state that a role is transitive and irreflexive, obtaining a strict partial order, this role would violate the simplicity requirements of roles in OWL DL: an ontology containing both statements will be in OWL Full. We can decompose this role into a single-step irreflexive role and a transitive role that includes the single-step role to try to avoid the conflict with simple roles as seen in Figure 15, but the transitive role must also be irreflexive to correctly capture the order. In some cases, like Competency Questions 1-9, the transitivity alone will be sufficient, so this can be a useful if imprecise strategy.

It is also possible to create a non-strict partial order if we use boolean role restrictions [27]. An example of this is reproduced in Figure 16. Unfortunately, boolean role constructors are only known to be decidable on simple roles, and our axioms require the use of functionality for identity and transitivity or irreflexivity in the already problematic strict partial order. Finally we find ourselves squarely in OWL Full with no obvious way to represent even a useful fragment of the expression. As such, the role representing non-strict partial orders may be helpful in axioms for modeling, but it will be incompatible with current OWL reasoners.

The potential use of orders in OWL warrants further investigation, since orders often have decidable properties and are convenient for modeling. Though it is unlikely that OWL can state all of the requirements for an order separately within the current semantics, based on what we have seen, perhaps future

$$\text{Reflexive}(\text{identity})$$
$$\text{Functional}(\text{identity})$$
$$\text{Irreflexive}(\text{directlyFollows})$$
$$\text{Irreflexive}(\text{inStrictPartialOrder})$$
$$\text{directlyFollows} \sqsubseteq \text{inStrictPartialOrder}$$
$$\text{directlyFollows} \circ \text{inStrictPartialOrder} \sqsubseteq \text{inStrictPartialOrder}$$
$$\exists(\text{inStrictPartialOrder} \cap \text{inStrictPartialOrder}^{-} \setminus \text{identity}).\top \sqsubseteq \bot$$
$$\text{inNonStrictPartialOrder} \equiv (\text{inStrictPartialOrder} \cup \text{identity})$$

Fig. 16. A non-strict partial order made from four roles

research will reveal that OWL can accommodate ordered roles in some other way.

7 Conclusions

We have presented a general ontology design pattern for trees together with an axiomatization which makes it possible to answer non-trivial competency questions as they arise in practice. We have also presented a list pattern derived from this tree pattern. We have furthermore discussed limitations of OWL for the modeling of trees, and have provided an alternative axiomatization for the more specific case that the tree is known to be n-bounded. Finally, ordered trees provided a useful extension to general trees as well as some reflections on orders and OWL.

Of course, our approach is still rather straightforward and there are cases where our model will not suffice. For example, in the application domain discussed in Section 2, it is often desirable to attach additional information to parent-child relationships (i.e., edges), e.g. temporal information. This cannot be done in our current model but would require a reification of the edges using established techniques, and of course this change may affect the treatment of our competency questions. This remains to be investigated.

Acknowledgements. Pascal Hitzler and Hilmar Lapp acknowledge support by the National Science Foundation (NSF) under awards OIA-2033521 "KnowWhereGraph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies", and DBI-1458484 "An Ontology-Based System for Querying Life in a Post-Taxonomic Age", respectively. Aaron Eberhart acknowledges support by the Air Force Office of Scientific Research under award number FA9550-18-1-0386. David Carral acknowledges support by the Deutsche Forschungsgemeinschaft (DFG) grant 389792660 as part of TRR 248 and Emmy Noether grant KR 4381/1-1.

References

1. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Mylopoulos, J., Reiter, R. (eds.) Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991. pp. 446–451. Morgan Kaufmann (1991), http://ijcai.org/Proceedings/91-1/Papers/069.pdf
2. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Boutilier, C. (ed.) IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 714–720 (2009), http://ijcai.org/Proceedings/09/Papers/124.pdf
3. Carral, D., Hitzler, P., Lapp, H., Rudolph, S.: On the ontological modeling of trees. In: Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., Hoekstra, R. (eds.) Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017. CEUR Workshop Proceedings, vol. 2043. CEUR-WS.org (2017), http://ceur-ws.org/Vol-2043/paper-01.pdf
4. Cellinese, N., Lapp, H.: An Ontology-Based system for querying life in a Post-Taxonomic age (2015), https://figshare.com/articles/An_Ontology_Based_System_for_Querying_Life_in_a_Post_Taxonomic_Age/1401984
5. Darwin, C.: On the Origin of Species. John Murray, London, 6 edn. (1859)
6. Dawson Jr., J.W.: The compactness of first-order logic:from Gödel to Lindström. History and Philosophy of Logic 14(1), 15–37 (1993)
7. Driskell, A.C., Ané, C., Burleigh, J.G., McMahon, M.M., O'meara, B.C., Sanderson, M.J.: Prospects for building the tree of life from large sequence databases. Science 306(5699), 1172–1174 (12 Nov 2004), http://dx.doi.org/10.1126/science.1102036
8. Dunn, C.W., Hejnol, A., Matus, D.Q., Pang, K., Browne, W.E., Smith, S.A., Seaver, E., Rouse, G.W., Obst, M., Edgecombe, G.D., Sørensen, M.V., Haddock, S.H.D., Schmidt-Rhaesa, A., Okusu, A., Kristensen, R.M., Wheeler, W.C., Martindale, M.Q., Giribet, G.: Broad phylogenomic sampling improves resolution of the animal tree of life. Nature 452(7188), 745–749 (Apr 2008), http://dx.doi.org/10.1038/nature06614
9. Felsenstein, J.: Phylogenies and the comparative method. Am. Nat. 125(1), 1–15 (1985), http://www.jstor.org/stable/10.2307/2461605
10. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, 2 edn. (4 Sep 2003)
11. Goloboff, P., Catalano, S., Mirande, J., Szumik, C., Arias, J., Källersjö, M., Farris, J.: Phylogenetic analysis of 73 060 taxa corroborates major eukaryotic groups. Cladistics 25(3), 211 (2009), http://dx.doi.org/10.1111/j.1096-0031.2009.00255.x
12. Hinchliff, C.E., Smith, S.A., Allman, J.F., Burleigh, J.G., Chaudhary, R., Coghill, L.M., Crandall, K.A., Deng, J., Drew, B.T., Gazis, R., Gude, K., Hibbett, D.S., Katz, L.A., Laughinghouse, 4th, H.D., McTavish, E.J., Midford, P.E., Owen, C.L., Ree, R.H., Rees, J.A., Soltis, D.E., Williams, T., Cranston, K.A.: Synthesis of phylogeny and taxonomy into a comprehensive tree of life. Proc. Natl. Acad. Sci. U. S. A. 112(41), 12764–12769 (13 Oct 2015), http://dx.doi.org/10.1073/pnas.1423041112
13. Hitzler, P., Krisnadhi, A.: On the roles of logical axiomatizations for ontologies. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns – Foundations and Applications, Studies on the Semantic Web, vol. 25, pp. 73–80. IOS Press (2016)

14. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation (11 December 2012), http://www.w3.org/TR/owl2-primer/
15. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press/Chapman & Hall (2010)
16. Huelsenbeck, J.P., Bollback, J.P., Levine, A.M.: Inferring the root of a phylogenetic tree. Syst. Biol. 51(1), 32–43 (Feb 2002), http://dx.doi.org/10.1080/106351502753475862
17. Jarvis, E.D., Mirarab, S., Aberer, A.J., Li, B., Houde, P., Li, C., Ho, S.Y.W., Faircloth, B.C., Nabholz, B., Howard, J.T., Suh, A., Weber, C.C., da Fonseca, R.R., Li, J., Zhang, F., Li, H., Zhou, L., Narula, N., Liu, L., Ganapathy, G., Boussau, B., Bayzid, M.S., Zavidovych, V., Subramanian, S., Gabaldón, T., Capella-Gutiérrez, S., Huerta-Cepas, J., Rekepalli, B., Munch, K., Schierup, M., Lindow, B., Warren, W.C., Ray, D., Green, R.E., Bruford, M.W., Zhan, X., Dixon, A., Li, S., Li, N., Huang, Y., Derryberry, E.P., Bertelsen, M.F., Sheldon, F.H., Brumfield, R.T., Mello, C.V., Lovell, P.V., Wirthlin, M., Schneider, M.P.C., Prosdocimi, F., Samaniego, J.A., Vargas Velazquez, A.M., Alfaro-Núñez, A., Campos, P.F., Petersen, B., Sicheritz-Ponten, T., Pas, A., Bailey, T., Scofield, P., Bunce, M., Lambert, D.M., Zhou, Q., Perelman, P., Driskell, A.C., Shapiro, B., Xiong, Z., Zeng, Y., Liu, S., Li, Z., Liu, B., Wu, K., Xiao, J., Yinqi, X., Zheng, Q., Zhang, Y., Yang, H., Wang, J., Smeds, L., Rheindt, F.E., Braun, M., Fjeldsa, J., Orlando, L., Barker, F.K., Jønsson, K.A., Johnson, W., Koepfli, K.P., O'Brien, S., Haussler, D., Ryder, O.A., Rahbek, C., Willerslev, E., Graves, G.R., Glenn, T.C., McCormack, J., Burt, D., Ellegren, H., Alström, P., Edwards, S.V., Stamatakis, A., Mindell, D.P., Cracraft, J., Braun, E.L., Warnow, T., Jun, W., Gilbert, M.T.P., Zhang, G.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science 346(6215), 1320–1331 (12 Dec 2014), http://dx.doi.org/10.1126/science.1253451
18. Krisnadhi, A., Hitzler, P.: Modeling with ontology design patterns: Chess games as a worked example. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns: Foundations and Applications, Studies on the Semantic Web, vol. 25, pp. 3–22. IOS Press, Amsterdam (2016)
19. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and rules. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P.F. (eds.) Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer (2011)
20. Maddison, W.P., Donoghue, M.J., Maddison, D.R.: Outgroup analysis and parsimony. Syst. Biol. 33(1), 83–103 (1 Mar 1984), https://academic.oup.com/sysbio/article-abstract/33/1/83/1669598/Outgroup-Analysis-and-Parsimony?redirectedFrom=fulltext
21. Martínez, D.C., Janowicz, K., Hitzler, P.: A logical geo-ontology design pattern for quantifying over types. In: Cruz, I.F., Knoblock, C.A., Kröger, P., Tanin, E., Widmayer, P. (eds.) SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012. pp. 239–248. ACM (2012)
22. Michael Keesey, T.: A mathematical approach to defining clade names, with potential applications to computer storage and processing. Zool. Scr. 36(6), 607–621 (Nov 2007), http://dx.doi.org/10.1111/j.1463-6409.2007.00302.x

23. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. J. Web Semant. 3(1), 41–60 (2005), https://doi.org/10.1016/j.websem.2005.05.001
24. O'Meara, B.C.: Evolutionary inferences from phylogenies: A review of methods. Annu. Rev. Ecol. Evol. Syst. 43(1), 267–85 (Nov 2011), http://dx.doi.org/10.1146/annurev-ecolsys-110411-160331
25. de Queiroz, K., Gauthier, J.: Phylogeny as a central principle in taxonomy: Phylogenetic definitions of taxon names. Syst. Biol. 39(4), 307–322 (1 Dec 1990), https://academic.oup.com/sysbio/article-abstract/39/4/307/1646987/Phylogeny-as-a-Central-Principle-in-Taxonomy
26. de Queiroz, K., Gauthier, J.: Phylogenetic taxonomy. Annu. Rev. Ecol. Syst. 23(1), 449–480 (1 Nov 1992), https://doi.org/10.1146/annurev.es.23.110192.002313
27. Rudolph, S., Krötzsch, M., Hitzler, P.: Cheap Boolean Role Constructors for Description Logics. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA. Lecture Notes in Computer Science, vol. 5293, pp. 362–374. Springer (2008)
28. Sereno, P.C.: The logical basis of phylogenetic taxonomy. Syst. Biol. 54(4), 595–619 (Aug 2005), http://dx.doi.org/10.1080/106351591007453
29. Shimizu, C., Hitzler, P., Horridge, M.: Rendering OWL in description logic syntax. In: Proceedings of the ESWC 2017 Poster and Demo Session (2017), to appear
30. Simpson, G.G.: Tempo and Mode in Evolution. Columbia University Press, New York (1949)
31. Simpson, G.G.: The Major Features of Evolution. Columbia University Press, New York (1953)
32. Smith, S.A., Beaulieu, J.M., Donoghue, M.J.: Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. BMC Evol. Biol. 9, 37 (11 Feb 2009), http://dx.doi.org/10.1186/1471-2148-9-37
33. Swofford, D.L., Olsen, G.J., Waddell, P.J., Hillis, D.M.: Phylogenetic inference. In: Hillis, D.M., Moritz, C., Mable, B.K. (eds.) Molecular systematics, pp. 407–514. Sinauer Associates, Inc., 2 edn. (1996), http://www.citeulike.org/group/1390/article/768694