# Reasoning about Taxonomies

By

David Michael Thau

B.S. Cognitive Science (University of California, Los Angeles) 1989
M.S. Psychology (University of Michigan, Ann Arbor) 1991
M.S. Computer Science (University of Michigan, Ann Arbor) 1993

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

---

Bertram Ludäscher (Chair)

---

Premkumar Devanbu

---

James F. Quinn

Committee in Charge
2010

David Michael Thau
June 2010
Computer Science

Reasoning about Taxonomies

**Abstract**

Taxonomically organized data pervade science, business, and everyday life. Unfortunately, taxonomies are often under-specified, or even inconsistent, limiting their utility in contexts such as data integration, information navigation, and autonomous agent communication. This work formalizes taxonomies and articulations (relationships between taxa in taxonomies) as first-order formulas. This formalization concretizes notions such as *consistency* and *inconsistency* of taxonomies and articulations between them, enables the derivation of new articulations based on a given set of taxonomies and articulations, and provides a framework for testing assumptions about under-specified taxonomies.

Given the typical intractability of reasoning with taxonomies and articulations, this research also investigates many optimizations: from those that reduce the search space, to those that leverage parallel processing, to those investigating logics more tractable than first-order logic (*e.g.*, monadic first-order logic, propositional logic, description logics, and subsets of the RCC-5 spatial algebra). Finally, in addition to reasoning with taxonomies and articulations, this research investigates how to merge taxonomies given articulations and how to merge data sets that have been annotated to aligned taxonomies. Critical to this research is the development of a framework for testing logics and supporting the development of taxonomies and articulations. This framework, CleanTax, has been implemented and has been used to study articulations between several large-scale biological taxonomies.

Dedicated to Kirsten Rose Menger-Anderson

# Acknowledgments

This work would not have been possible without the help of a great number of people. In roughly chronological order, I will start with my parents, Robert and Vera Thau, who instilled in me interests in computer science, human learning and classification, and natural history. Jumping ahead in time, I'd like to thank my undergraduate advisor Keith Holyoak, whose work on concept mapping in analogy informed my research, and Douglas Medin, whose research in the psychological underpinnings of human categorizations of life forms provided me with an early nudge in the direction of this dissertation. The direct roots of the research described here were planted at the All Species Foundation, and I would like to thank Kurt Bollacker, Kevin Kelly, and Ryan Phelan for involving me in that project. While there, I met David Vieglais, Stan Blum, Brian Fisher, Robert A. Morris, John Wieczorek, and Jim Beach, all of whom have been a continual source of inspiration, support and assistance. Through this set of people, I became involved in the Science Environment for Ecological Knowledge (SEEK) project, where I met Bertram Ludäscher, Shawn Bowers, Bob Peet, Matt Jones, Aimee Stewert, Jessie Kennedy, Mark Schildhauer, Nico Franz, Chad Berkley, Deana Pennington, and numerous other fantastic people who unwittingly convinced me to return to graduate school to focus on the work presented here. While in graduate school, I received a great deal of assistance and support from the members of the Data and Knowledge Systems lab and other students in the department, most notably Daniel Zinn, Tim McPhillips, Sean Riddle, Manish Anand, Carlos Rueda, Till Stegers, Balaji Venkatachalam, and Ananya Das. I also had the pleasure of receiving advice from Richard Waldinger, Todd J. Green, David Maier, Ellen Spertus, Eve Menger-Hammond, and the esteemed members of my dissertation committee: Bertram Ludäscher, Premkumar Devanbu, and Jim Quinn. I would especially like to thank my frequent co-author Shawn Bowers, and my advisor Bertram Ludäscher both of whom spent incredible amounts of time and effort helping me along with the work in this thesis. Finally, I'd like to

acknowledge the support, patience, and frequent assistance of my wonderful wife, Kirsten Menger-Anderson.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Humans classify, and taxonomies are one of the most natural forms of classification. Taxonomies pervade our lives, from the Dewey Decimal system of book classification, to business org-charts, to biological taxonomies used to define the tree of life. Taxonomies are prevalent in science and engineering, where data are often organized hierarchically. Because taxonomies are "views" on data, different taxonomies organizing the same set of data often arise due to changing domain information or differing expert opinion. These varying taxonomies can make data integration difficult, especially when data are distributed or organized using different but related schemas. Integrating data organized by alternative taxonomies requires, in addition to the given taxonomies, a set of articulations indicating how the concepts (taxa, classes) in the different taxonomies relate to one another. A set of taxonomies and articulations among their concepts is called an *alignment*. The process of discovering an alignment given a set of taxonomies is called the *taxonomy alignment problem.*

Taxonomies and articulations between them have been created in many domains. For example, in biology, a recent analysis of treatments of the plant genus Ranunculus [Pee05] considered 9 taxonomies, covering 654 concepts (called *taxa* in biology) and 704 articula-

tions. In education, [MW82] compared taxonomies of postsecondary-education institutions from three different institutions, judging some taxonomies as "more efficient" classifiers. In library sciences, a number of mappings between the Dewey Decimal and Library of Congress taxonomies for literature subjects have been created (*e.g.*, [wik07, que03]).

Taxonomies can be large, potentially necessitating a great number of articulations. For example, [AGY05] compares three Web directories (Google, Looksmart, and Yahoo) to analyze taxonomy alignment systems. Each directory has hundreds of thousands of nodes. A node-by-node pairwise comparison of Google and Looksmart, for example, yields almost 300 billion potential articulations.

The research described here addresses reasoning about taxonomies and articulations drawn between concepts in multiple taxonomies. Topics within this focus range from investigations into modeling taxonomies and articulations, the use of automatic reasoners to detect inconsistencies and uncertainty introduced by articulations, optimizations for calculating the deductive closure of a set of taxonomies and articulations, mechanisms for merging taxonomies, and methods for merging datasets that have been annotated to aligned taxonomies.

## 1.2 Scenarios

### 1.2.1 Data Integration

Consider a biologist faced with combining species occurrence data from multiple studies. Species are organized hierarchically, and the definitions of species names change over time, so datasets using different biological taxonomies may classify species differently [KKP05]. If the data are classified using well-known taxonomies, and an expert has indicated how concepts in the various taxonomies relate, it may be possible to automatically integrate the data. Before this is possible, the articulations between the taxonomies must be drawn.

To create the articulations, a metadata curator must consider multiple taxonomies and draw the articulations between their concepts. While creating articulations between these

| Species | Count | Site | Transect | Date |
|---|---|---|---|---|
| *Ranunculus arizonicus* (Benson, 1948) | 30 | 1 | 1 | January 1, 2005 |
| *Ranunculus acriformis* (Benson, 1948) | 12 | 1 | 1 | January 1, 2005 |
| *Ranunculus arizonicus* (Benson, 1948) | 8 | 1 | 2 | January 1, 2005 |

Table 1.1: Dataset with abundance counts by observer $O_1$

| Species | Count | Site | Transect | Date |
|---|---|---|---|---|
| *Ranunculus arizonicus* (Kartesz, 2004) | 6 | 1 | 1 | June 22, 2005 |
| *Ranunculus aestivalis* (Kartesz, 2004) | 18 | 1 | 1 | June 22, 2005 |
| *Ranunculus glabberimus* (Kartesz, 2004) | 3 | 1 | 2 | June 22, 2005 |

Table 1.2: Dataset with abundance counts by observer $O_2$

taxonomies, the curator will want to know, (i) when a newly proposed articulation leads to an impossible situation, (ii) if so, why, and how the inconsistency may be repaired, (iii) if the proposed articulations entail other previously unknown articulations, and (iv) if an unexpected articulation is entailed, how it was derived.

The answers to these questions depend in part on the taxonomies being compared. Unfortunately, taxonomies are frequently under-specified, described only by the subsumption relationships of the concepts. Taxonomies often carry additional unstated constraints, such as that taxa subsumed by a common parent should be disjoint. These constraints may impact the logical consistency of articulations, as well as the entailment of additional articulations. In general, an articulator will prefer articulations that minimize ambiguity, choosing unambiguous articulations such as "concept A and concept B are congruent" over ambiguous ones, such as "concept A and concept B are either congruent, or A is a subset of B." Given this goal, the articulator may want to assume that the taxonomies in question follow certain constraints. Once the articulations have been created, they may be checked for validity and stored for later use.

**Example 1.1.** Imagine attempting to predict the geographic distribution of a taxon, for example the species *Anhinga melanogaster*. The Global Biodiversity Information Facility (GBIF) provides a good initial source of locality information for this taxon. Figure 1.1 (a)

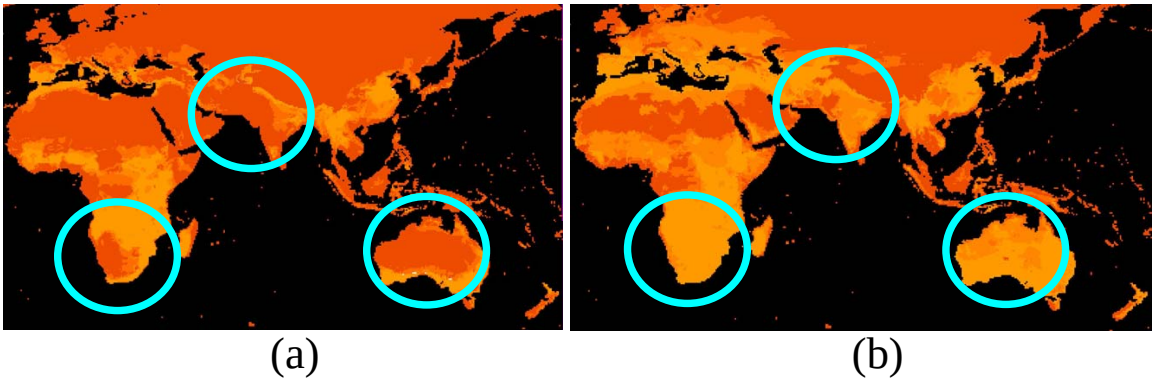



Figure 1.1: Predicted distribution for *Anhinga melanogaster* according to (a) *Clement's Birds of the World, 4th edition* [Cle91] and (b) *Clement's Birds of the World, 5th edition* [Cle01]

shows a species range prediction for *Anhinga melanogaster* generated using 100 iterations of the GARP algorithm run within a Kepler workflow [LAB+06]. Lighter shades represent greater probability of presence.

Note that there appears to be a low probability of finding *Anhinga melanogaster* in South Africa, central Australia, western India, Pakistan and Afghanistan. However, a recent version of Clement's Birds of the World lumps the species *Anhinga rufa* and *Anhinga novaehollaniae* into *Anhinga melanogaster*. An adherent to this taxonomy would merge the locality data of these three species, resulting in the species range prediction shown in Figure 1.1 (b). Knowing the relationships between different versions of the same taxon has a clear impact on predictions about that taxon.

**Example 1.2.** Consider two datasets involving abundances of various species of the plant genus *Ranunculus* (which contains the buttercups), in two transects of a given locality. The first dataset (Table 1.1) represents observations of *Ranunculi* taken by a person $O_1$ who used a field guide based on Benson, 1948. The second dataset (Table 1.2) represents observations of *Ranunculi* taken in the same location, by a different person $O_2$, six months later. Observer $O_2$ used a field guide based on Kartesz, 2004. Assume that both studies attempted to document all species of *Ranunculus* observed in each locality.

We can ask a number of queries over these datasets, *e.g.*, *"What was the average number of Ranunculus arizonicus observed in Transect 1 of LTER Site 1?"* Observer $O_1$ (using Benson, 1948) found 30 examples of *Ranunculus arizonicus* in Transect 1, while $O_2$ (using Kartesz, 2004) found 6 examples of *Ranunculus arizonicus* in the same spot. Given no information about the relationship between Benson's and Kartesz's concepts of *R. arizonicus*, we cannot safely average over these two datasets:

(i) It may be that both $O_1$ and $O_2$ would have agreed that every observed *R. arizonicus* was in fact an *R. arizonicus*. We could assume this if Benson's and Kartesz's concepts of *R. arizonicus* were known to be equivalent. In other words, every plant classified by Benson, 1948 as *R. arizonicus* would have also been classified by Kartesz, 2004 as *R. arizonicus* and vice versa. In this case, the answer to our query is that on average 18 ($= \frac{30+6}{2}$) *R. arizonicus* were seen in transect 1.

(ii) Alternatively, Benson's and Kartesz's concepts of *R. arizonicus* could be so different that $O_2$ would have classified every *R. arizonicus* of $O_1$ as some other species, and similarly, $O_1$ would have classified every *R. arizonicus* of $O_2$ as another species. In this case, the abundance data for *R. arizonicus* in the two datasets refer to two distinct plant species, rendering an average of the two abundances meaningless.

(iii) Finally, assume that based on scientific literature, we may ascertain that all examples classified as *R. arizonicus* by Kartesz, 2004 would have also been classified as *R. arizonicus* by Benson, 1948, but *not* vice versa, denoted[1] *R. arizonicus*$^{\text{K04}} \subsetneq$ *R. arizonicus*$^{\text{B48}}$. Furthermore, assume there are no other types of *Ranunculus* considered by Kartesz, 2004 to be *R. arizonicus*. In this case, we could consider the numbers in the two datasets to be comparable according to the definition of Benson, 1948. Thus, we could answer the query by concluding that on average 18 examples of *R. arizonicus*$^{\text{B48}}$ were spotted in 2005. However, we cannot state an average in terms of the Kartesz, 2004 definition of *R. arizonicus* because some of the 30 *R. arizonicus*$^{\text{B48}}$ observed by $O_1$ may not be considered *R. arizonicus*$^{\text{K04}}$.

[1] The superscripts K04 and B48 indicate the authorities Kartesz, 2004 and Benson, 1948, respectively.

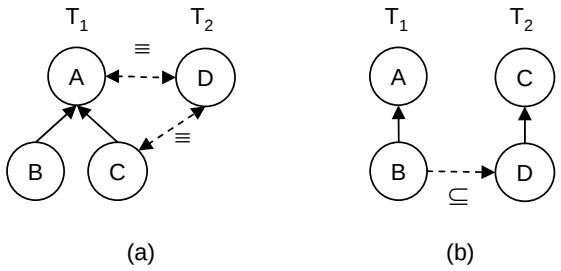



Figure 1.2: Simple examples of articulation inconsistency (a) and uncertainty (b). Articulations between taxonomies are marked with a dashed line.

Clearly, if we do not know the relationship between Benson's and Kartesz's concepts of *R. arizonicus*, the only accurate answer to the query is that the average number is uncertain because the observers collecting the data sets used different identification guides. However, knowing the relationship between these concepts affords a more precise answer.

### 1.2.2 Metadata Curation

Imagine an expert creating articulations between concepts in a pair of large taxonomies. Given a set of assumptions about the taxonomies, it can be quite easy to state articulations that are logically impossible. For example, if all concepts are assumed to contain at least one instance, and each parent concept cannot contain any instances not also contained in one of its child concepts, and sibling concepts contain disjoint instances, then the articulations shown in Figure 1.2 (a) are impossible. This may be seen by positing an instance of concept $B$. Any instance of $B$ must also be an instance of $A$. The equivalences between concepts $A$, $C$, and $D$ imply that any instance of $A$ must also be an instance of $C$. This means that the posited instance is a member of both $B$ and $C$ concepts, violating sibling disjointness. A metadata curator would want to know if he or she has created an inconsistent situation, and if so, what caused the inconsistency. Knowing the cause of an inconsistency, a metadata curator would probably also want to know how to remove the inconsistency without introducing other inconsistencies.

In a somewhat different scenario, a metadata curator might create articulations that

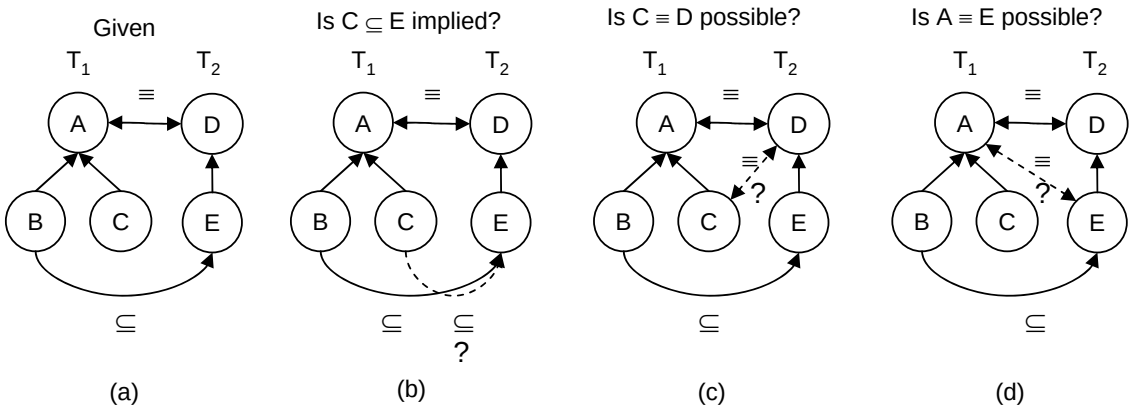



Figure 1.3: Possible questions about articulations between taxonomies

leave some uncertainty about the relationship between various concepts in the articulated hierarchy. For example, in Figure 1.2 (b), the relationship between taxa A and C is unclear. The two concepts may contain the same instances, or one may contain a subset of the other, or the concepts may overlap. A metadata curator will probably want to know when such uncertainty is introduced, why it is introduced, and how it may be removed.

### 1.2.3 Taxonomy Merging

Taxonomies need to be merged whenever organizations combine. For example, many businesses are now filing financial statements to their home governments using the Extensible Business Reporting Language (XBRL) [xbr] format. XBRL is an XML-based standard that contains taxonomies for financial reporting. XBRL was designed for businesses to create their own taxonomies, and different countries have embraced different standard taxonomies within the XBRL framework. When corporations merge, the resulting company will want access to the financial statements of the child corporations. This might require the creation of merged XBRL taxonomies.

### 1.2.4 Inference

Given some articulations, a metadata curator might want to know what other articulations are implied. This serves two purposes. First, automatic inference of articulations means less work for the curator. Second, surprising articulations may be a cause for celebration, if something new has been learned, or a cause for concern, if an unexpected articulation indicates an error in other articulations or in the taxonomies themselves.

Automatic inference of articulations depends on the taxonomies, a starting set of articulations, and additional assumptions about how taxa in the taxonomies relate. Consider the abstract taxonomies in Figure 1.3. Figure 1.3(a) depicts two taxonomies $T_1$ and $T_2$ and a pair of articulations. In $T_1$, $B$ and $C$ are child taxa of $A$, and in $T_2$, $E$ is a child of $D$. The articulations use set-theoretic notation to state that $A$ and $D$ are equivalent ($A \equiv D$ means all elements of $A$ are also elements of $D$ and vice versa), and $B$ is a subset of $E$ ($B \subseteq E$ means all elements of $B$ are also elements of $E$, but not necessarily the other way around). Given these taxonomies and the relations between them can we deduce that $C \subseteq E$ is implied (Figure 1.3(b))? Is it consistent to assert $C \equiv D$, or $E \equiv A$ (Figures 1.3(c) and 1.3(d), respectively)? The answers to these questions depend on various (sometimes latent) assumptions about $T_1$ and $T_2$. Once we spell out these additional taxonomic constraints in logic, all such questions have unambiguous answers (see Appendix A.4.1).

## 1.3 Domain Description

Traditionally, taxonomies have been defined as a partial ordering of concepts where the ordering relation denotes some sort of "inclusion" relation [Bra83]. The partial ordering relation is often called an `isa` relation, and represents a rule of the form: A `isa` B means that if instance x is an example of A, then x is also an example of B. In biological taxonomies, this translates to rules like, "if Fido is an instance of Canis lupus, then Fido is an instance of Canis." In a phylogeny, the rule might be "if Fido is an instance of the things descended from ancestor A, then it is also an instance of things descended from ancestor B." In a

business org-chart, the rule might be "if employee x is a member of the software engineering department, then employee x is also a member of the engineering department."

This definition is very general. Partial orders may be strict (the ordering relation is irreflexive, asymmetric, and transitive) or non-strict (the ordering relation is reflexive, transitive, and antisymmetric). Some taxonomies permit multiple inheritance while others do not. In some taxonomies, child concepts partition their parents, while in others, parent concepts may contain instances not contained in their children.

In addition to being too general, the definition of a taxonomy as a partial order does not describe how taxonomies are actually defined. Taxonomies may be defined entirely as graphs, with edges representing inclusion relations and nodes representing taxonomic concepts. In other cases, such as in Formal Concept Analysis (FCA) [GW99], taxonomies are defined using the properties of the instances of the concepts. In yet other cases, taxonomies are defined using characteristics of the concepts, without reliance on instances. Sometimes a combination of these occurs [CdQ06].

When given two or more taxonomies organizing similar data, it is natural to wonder how the taxonomies relate. One method of comparing taxonomies is to describe relationships between concepts in each taxonomy (the articulations). The vocabulary used to describe these articulations can be very simple or quite complex, depending on the language used to express the articulations, and the taxonomies themselves.

In summary, in a given setting, the taxonomies and articulations may be syntactically heterogeneous (not expressed in the same language), terminologically heterogeneous (the "same" entities might have different names), and conceptually heterogeneous [BBG01] (*e.g.*, the taxonomies about the "same" concepts might cover different parts of the world).

## 1.4 Outstanding Problems

Given this context of taxonomies and articulations between them, a number of questions arise. These questions may be thought of as use cases for a system that helps build and

work with taxonomies and articulations. These questions are stated broadly here, but will be formalized later. The questions below are loosely grouped by topic. The chapters of this dissertation that address the given group of questions follow the group heading.

**Questions about Representation and Reasoning** [Chapter 3, Chapter 4, Chapter 5]

**Question 1.3** (Representation)**.** How are taxonomies and relations between entities in different taxonomies represented?

**Question 1.4** (Representation of uncertainty)**.** How should uncertainty in the relationship between two concepts be represented?

**Question 1.5** (Consistency of taxonomies)**.** Given a formal definition of taxonomy, is a given set of concepts and relations a valid taxonomy?

**Question 1.6** (Models)**.** Does a set of instances and information about how those instances are sorted into the concepts of a taxonomy describe a model that satisfies a given taxonomy?

**Question 1.7** (Consistency of alignment)**.** Given a pair of consistent taxonomies, and relations between their concepts, are there any contradictory assertions?

**Question 1.8** (Inference)**.** Are any unstated relations implied?

**Questions about Explanations** [Chapter 6, Chapter 7, Chapter 9]

**Question 1.9** (Explanation of Inconsistency)**.** From where do contradictions arise?

**Question 1.10** (Explanations of discovered relations)**.** If unstated relations are discovered, how are they derived?

**Question 1.11** (Interestingness)**.** Which discovered relations are interesting, and how does one define interestingness?

**Question 1.12** (Minimality)**.** Are the given relations within a taxonomy or between articulated taxonomies a minimal set, or may some be removed while entailing the same relations?

**Question 1.13** (Maximal/minimal consistent/inconsistent subset)**.** What are the maximal consistent or minimal inconsistent subsets of concepts and relations in single taxonomies or articulated taxonomies?

**Question 1.14** (Merge)**.** Is there a single canonical representation of the combination of two or more taxonomies, given relationships between them?

**Question 1.15** (Data Integration)**.** Can a taxonomic alignment be leveraged to merge sets of data that draw their terms from the taxonomies?

## 1.5 Current Solutions

The questions above span several topics: representation, taxonomy alignment, taxonomy merging, and data integration. Each of these topics has received attention in a number of different domains, and literature reviews for each topic will be provided in the relevant chapters. Here, however, I will describe some of the key differentiators between different approaches to alignment, and merging problems in general.

The first basic differentiator is the type of information used in the aforementioned operations. Some techniques rely on the existence of instances [SM01a], versus those that do not take instances into account, such as PROMPT [NM03]. Some techniques rely on a lexical analysis of the names of entities [DR02], while others focus on structural elements such as the relations between concepts [GYS07]. Like [GYS07], the current research focuses entirely on the structure of the relations between concepts in the taxonomies being aligned. This focus may be seen as a point of departure from which investigations into the effect of instances and lexical similarities may be investigated.

Another key differentiator between solutions involves the languages used to express the data. For example, in [ST05] ontologies are represented in a modified description logic, and relations between concepts in the ontologies may be either subsumption, equivalence, or disjointness. In [GYS07], on the other hand, ontologies are represented in propositional logic, and relations between concepts are modeled as implication, equivalence, and disjointness.

A third example is [MHH+01], which models relational databases and XML documents in a nested relational model and supports arbitrary *n:m* transformations between concepts in the inputs. In this dissertation, taxonomies will be represented in a variety of logics, including first-order logic, description logic, the region connection calculus, and propositional logic.

## 1.6 Goals and Contributions

The goal of the proposed research is to formally define taxonomies and articulations between them, and describe algorithms for performing a number of operations on multiple articulated taxonomies. Among these operations are inferring unstated articulations, explaining the inferences, explaining inconsistencies in taxonomies and articulations, quantifying uncertainty in articulated taxonomies and aiding in the reduction of that uncertainty, merging articulated taxonomies, and merging data sets that have been annotated to aligned taxonomies. In order to support these operations, a number of logics (*e.g.*, first-order logic, description logic, propositional logic) will be investigated in terms of their expressiveness in this context and the efficiency of performing the operations. As taxonomies can be quite large, most of the operations will require optimizations. To make a system implementing the operations useful, visualizations of the taxonomies, articulations, and products of the operations (such as a merged taxonomy) will be necessary. Finally, the system must be implemented, and three different implementations are described here: a traditional object-oriented implementation; an implementation like the first, but optimized for running in a Grid environment; and a workflow-based implementation.

## 1.7 Thesis Structure

This dissertation is organized as follows:

Chapter 2 introduces ideas and notations that will hold throughout the dissertation.

Chapter 3 formalizes many of the notions used throughout the dissertation, including taxonomies, articulations, and alignments. Chapter 4 instantiates the formalization of taxonomy alignment using a subset of first-order logic called monadic first-order logic and applies the formalization to a real-world data set. Chapter 5 discusses optimizations useful for performing the tasks in Chapter 4, including optimizations to reduce the number of proofs necessary, and optimizations involving less expressive logics. Chapter 6 discusses how articulated taxonomies may be merged together to form a unified taxonomy useful in many data integration contexts. Chapter 7 describes algorithms for merging presence/absense data sets that use vocabularies drawn from aligned taxonomies. Chapter 8 describes three implementations of the CleanTAX framework: a traditional object-oriented implementation; an implementation like the first, but optimized for running in a Grid environment; and a workflow based implementation. Chapter 9 sums up the contributions of this work and outlines future work on explaining inferences and inconsistencies, repairing inconsistencies, judging the interestingness of new articulations, and reducing uncertainty in alignments.

# Chapter 2

# Preliminaries

This chapter describes the notation and foundational terms used throughout the rest of this work. It is meant primarily as a reference.

## 2.1 Notation

This dissertation uses the following conventions. All of the terms below are defined in the subsequent definition section.

| | |
|---|---|
| Constants | $a, b, c, \ldots$ |
| Variables | $x, y, z, \ldots$ |
| Nodes, predicates, tuples | $N, M, \ldots$ |
| Sets | $\mathbf{N}, \mathbf{M}, \ldots$ |
| First-order formulas | $\phi, \psi, \ldots$ |
| Interpretations, structures | $\mathcal{I}, \mathcal{J}, \ldots$ |
| Languages | $\mathcal{L}, \mathcal{M}, \ldots$ |

## 2.2 Definitions

### 2.2.1 Sets, Partial Orders, Lattices, and Power Sets

**Sets.** The following definitions are largely taken from [DP02]. A *set* $\mathbf{M}$ is a collection of distinct objects $m$ (which are called the *elements* or *instances* of $\mathbf{M}$). Sets may be defined *extensionally* using a list of instances (called an *extent*), or *intensionally* using a rule or semantic description (*e.g.*, the set of prime numbers). The binary relation $\in$ is used to indicate that an element is "in" a set or is a "member" of the set. One set $\mathbf{N}$ is a *subset* of another set $\mathbf{M}$ ($\mathbf{N} \subseteq \mathbf{M}$) if for all $x$, if $x \in \mathbf{N}$ then $x \in \mathbf{M}$.

**Partial orders.** Given a set $\mathbf{M}$, a *partial order* on $\mathbf{M}$ is a binary relation $\leqslant$ such that, for all $x, y, z \in \mathbf{M}$: (i) $x \leqslant x$ (reflexivity), (ii) $x \leqslant y$ and $y \leqslant x$ implies $x = y$ (antisymmetry), and (iii) $x \leqslant y$ and $y \leqslant z$ implies $x \leqslant z$ (transitivity). An *ordered set* $(\mathbf{M}, \leqslant)$, is a set equipped with a relation ordering the set. If $\mathbf{M}$ is an ordered set, we say that $\mathbf{M}$ has a *bottom* if there exists a $\bot \in \mathbf{M}$, which has the property $\bot \leqslant x$ for all $x \in \mathbf{M}$. Similarly, $\mathbf{M}$ has a *top* if there exists a $\top \in \mathbf{M}$ with the property $x \leqslant \top$ for all $x \in \mathbf{M}$.

**Supremum, infimum, meet, and join.** Let $\mathbf{M}$ be an ordered set and let $\mathbf{N} \subseteq \mathbf{M}$. An element $m \in \mathbf{M}$ is an *upper bound* of $\mathbf{N}$ if $n \leqslant$ m for all $n \in \mathbf{N}$ and $\mathbf{N}^u$ is the set of all upper bounds of $\mathbf{N}$ (also called $\mathbf{N}$ *upper*). If $\mathbf{N}^u$ has a least element $x$ then $x$ is the *least upper bound* (or *supremum*) of $\mathbf{N}$. The terms *lower bound*, $\mathbf{N}$ *lower* ($\mathbf{N}^l$) and *greatest lower bound* (or *infimum*) are defined dually. If two elements, $x, y \in \mathbf{M}$ have a least upper bound, we write $x \vee y$, read as "x *join* y." Similarly, if two elements, $x, y \in \mathbf{M}$ have a greatest lower bound, we write $x \wedge y$, read as "x *meet* y." If a least upper bound for a set $\mathbf{M}$ exists, we write $\bigvee \mathbf{M}$, read "the join of $\mathbf{M}$." Similarly, if a greatest lower bound for a set $\mathbf{M}$ exists, we write $\bigwedge \mathbf{M}$, read "the meet of $\mathbf{M}$."

**Lattices.** A *lattice* is a non-empty ordered set in which for all $x, y \in \mathbf{M}$, $x \vee y$ and $x \wedge y$ exist. If every subset $\mathbf{N}, \mathbf{N} \subseteq \mathbf{M}$ has a meet and a join, we say the set $\mathbf{M}$ is a *complete*

*lattice.*

**Power sets.** Given a set $\mathbf{M}$, the power set $\mathcal{P}(\mathbf{M})$ (also denoted by $2^{\mathbf{M}}$) is the set of all subsets of $\mathbf{M}$. Ordering the elements of a power set with respect to inclusion results in a complete lattice.

### 2.2.2 Graphs and Trees

**Graphs and edges.** A *directed graph* consists of a finite nonempty set $\mathbf{N}$ of nodes and a finite set $\mathbf{E} \subseteq \mathbf{N} \times \mathbf{N}$ of directed edges. A directed edge $\mathbf{E} = (N, M)$ is an ordered pair of nodes, where $N$ is said to be the *predecessor* of $M$ and $M$ is the *successor* of $N$. A directed edge $\mathbf{E} = (N, M)$ is also said to be *incident* with nodes $N$ and $M$.

**Walks and paths.** A *walk* from node $N_i$ to node $N_j$ in a graph is an alternating sequence $[N_i, E_{i+1}, N_{i+1}, E_{i+2}, \ldots, N_{j-1}, E_j, N_j]$ of nodes and edges in the graph such that $E_k = (N_{k-1}, N_k)$ $for$ $k = i+1, \ldots, j$. A walk is *closed* if its first and last nodes are the same, and *open* if they are different. An open walk is also called a *path.*

**Connected and acyclic graphs.** A graph $G = (\mathbf{N}, \mathbf{E})$ is *connected* if for every pair of nodes $N, M \in \mathbf{E}$ there is a walk between $N$ and M. A graph is *acyclic* if no node $N$ takes part in a closed walk.

**Trees and roots.** A graph is *rooted* if there is a distinguished node $R \in \mathbf{N}$ called the root of the graph such that for all nodes $N \in \mathbf{N}$ there is a path in G from the root $R$ to the node $N$. A *tree* is a connected, acyclic graph.

### 2.2.3 Formal Languages

A formal language is described by a *syntax* and a *semantics.* The syntax of the language defines a set of *well formed formulas* (wffs, often abbreviated as just *formulas*) that are legal in that language. The wffs are formed by defining a set of symbols, which define the

*alphabet* of the language, and a set of *formation rules*, which describe how the symbols may be combined to form formulas. The semantics provides a meaning for those formulas.

**Example 2.1** (A simple formal language)**.** As an example, here is a simple formal language $\mathcal{L}_0$

**Syntax.** The language $\mathcal{L}_0$ is built from an *alphabet* consisting of (i) a set of *variables* $\mathbf{V} = \{x, y, z, \dots\}$, (ii) a single *connective* between formulas $\rightarrow$, and (iii) a *signature* $\mathbf{S} = \mathbf{R} \cup \mathbf{C}$, involving sets of *predicate symbols* $\mathbf{R}$ $(R_1, R_2, \dots)$ and *constants* $\mathbf{C}$ $(c_1, c_2, \dots)$. In $\mathcal{L}_0$ each $R \in \mathbf{R}$ has an *arity* of 1. However, in most formal languages, each $R \in \mathbf{R}$ will have a unique *arity* $\geq 1$.

The set $\mathbf{T}$ of *terms* is the least set such that $\mathbf{V}, \mathbf{C} \subseteq \mathbf{T}$ (constants and variables are terms).

A $\mathcal{L}_0$ *formula* is either an *atomic formula* $R(t)$, with unary $R \in \mathbf{R}$ and $t \in \mathbf{T}$, or of the form $(\varphi \rightarrow \psi)$, where $\varphi, \psi$ are $\mathcal{L}_0$ formulas. Parentheses may be omitted when clear from the context.

**Semantics.** Fix a signature $\mathbf{S} = \mathbf{R} \cup \mathbf{C}$. A first-order *structure* $\mathcal{I} = (\mathbf{D}, I)$ for $\mathbf{S}$ consists of a *domain* $\mathbf{D}$ and a mapping $I$, assigning to every constant $c \in \mathbf{C}$, and unary predicate symbol $R \in \mathbf{R}$, a domain element $c^I$, and a unary predicate $R^I \subseteq \mathbf{D}$, respectively. Since $I$ interprets (*i.e.*, assigns meaning to) all symbols in $\mathbf{S}$, terms and formulas over $\mathbf{S}$ can be *evaluated* under $\mathcal{I}$, provided we also map free variables to domain elements via a *variable assignment* $\beta : \mathbf{V} \rightarrow \mathbf{D}$. Let $\mathfrak{I} = (\mathcal{I}, \beta)$ be an *interpretation*, *i.e.*, a first-order structure $\mathcal{I}$ with variable assignment $\beta$. Formula evaluation is defined inductively as a *satisfaction relation* $\mathfrak{I} \models \varphi$ ("$\mathfrak{I}$ *satisfies* $\varphi$", "$\mathfrak{I}$ *is a model of* $\varphi$", "$\varphi$ *holds in* $\mathfrak{I}$"):

$$\begin{aligned} \mathfrak{I} \models R(t) \quad & \text{iff} \quad \mathfrak{I}(t) \in R^I \\ \mathfrak{I} \models \varphi \rightarrow \psi \quad & \text{iff} \quad \mathfrak{I} \models \varphi \text{ implies } \mathfrak{I} \models \psi \end{aligned}$$

A formula $\varphi$ without free variables is called a *sentence* or *constraint*, and corresponds

to a yes/no (*boolean*) query. In this case, we use $\mathcal{I}$ instead of $\mathcal{J}$ because the variable assignment $\beta$ is no longer needed. Let $\Phi$ be a set of constraints. We write $\mathcal{I} \models \Phi$ if $\mathcal{I} \models \varphi$ for all $\varphi \in \Phi$ and say "$\mathcal{I}$ *is a model of* $\Phi$." We write $\Phi \models \varphi$ if every model of $\Phi$ is also a model of $\varphi$, *i.e.*, $\varphi$ is a (logical) *consequence* of $\Phi$.[1]

**Automated Reasoning.** The semantic consequence relation $\Phi \models \varphi$ can be "mechanized" using the rules of a *calculus*. A calculus is based on a *provability relation* $\Phi \vdash \varphi$, stating that $\varphi$ can be derived (formally proven) from the formulas in $\Phi$ and the derivation rules of the calculus. $\Phi$ is called *consistent* if there is no formula $\varphi$ such that both $\Phi \vdash \varphi$ and $\Phi \vdash \neg\varphi$; otherwise $\Phi$ is *inconsistent*. A calculus is *sound* if $\Phi \vdash \varphi$ implies $\Phi \models \varphi$ (everything that can be derived is a consequence), and *complete* if $\Phi \models \varphi$ implies $\Phi \vdash \varphi$ (every consequence can be derived).

[1] Note the difference between $\mathcal{I} \models \varphi$ and $\Phi \models \varphi$: the former is the satisfaction relation between a structure (database instance) $\mathcal{I}$ and a formula (query) $\varphi$; the latter is the consequence relation, stating that *all* structures $\mathcal{I}$ which satisfy $\Phi$ also satisfy $\varphi$. Thus, $\mathcal{I} \models \varphi$ is also called formula evaluation (given $\mathcal{I}$), while the $\Phi \models \varphi$ involves "reasoning" (independent of $\mathcal{I}$).

# Chapter 3

# Formal Modeling of the Domain

This chapter lays out the formal foundation for the chapters to follow. It contains definitions of the terms that will be used throughout the dissertation, and gives formal signatures for many of the operations that will later be implemented. The chapter concludes by showing how the terms and operations defined here may be applied to answer many of the questions in section 1.4.

## 3.1 Definitions

This section formally defines the terms used in this dissertation.

**Definition 3.1** (ISA Hierarchy)**.** An *isa hierarchy* is a pair $H = (\mathbf{N}, \leq_{isa})$ consisting of a set of *taxa* $\mathbf{N}$ (singular *taxon*) and a partial ordering relation $\leq_{isa}$ (called a *hierarchical relation*). The partial ordering relation, under a given interpretation, constrains the interpretation of the taxa in $\mathbf{N}$. These constraints are called *hierarchical constraints*. An individual hierarchical constraint may be represented as a pair $(N_1, N_2)$ meaning $N_1 \leq_{isa} N_2$. The first element of this pair is called the *predecessor* or *child* and the second is called the *successor* or *parent*.

**Example 3.2** (Example `isa` Hierarchy)**.**

$H = (\mathbf{N}, \leq_{isa})$

$\mathbf{N} = \{\mathbf{Animal}, \mathbf{Mammal}, \mathbf{Fish}, \mathbf{Dog}, \mathbf{Trout}\}$

$\leq_{isa} = \{(\mathbf{Mammal}, \mathbf{Animal}), (\mathbf{Fish}, \mathbf{Animal}), (\mathbf{Dog}, \mathbf{Mammal}), (\mathbf{Trout}, \mathbf{Fish})\}$

Given this set of taxa and hierarchical constraints, we can ask whether a given interpretation $\mathcal{I}$ satisfies the `isa` hierarchy $H$. Consider the following interpretation $\mathcal{I}$ of the nodes in $\mathbf{N}$ where a, b, and c are "real world" specimens: $Animal^{\mathcal{I}} = \{a, b, c\}, Mammal^{\mathcal{I}} = \{a, b\}, Fish^{\mathcal{I}} = \{c\}, Dog^{\mathcal{I}} = \{c\}, Trout^{\mathcal{I}} = \{a, b\}$. In order for the interpretation to satisfy the `isa` hierarchy, the $\leq_{isa}$ relation must hold. However, the given interpretation violates the hierarchical constraints $(\mathbf{Dog}, \mathbf{Mammal})$ and $(\mathbf{Trout}, \mathbf{Fish})$. For example, $\mathbf{Dog}^{\mathcal{I}} = \{c\}$ and $\mathbf{Mammal}^{\mathcal{I}} = \{a, b\}$ and $\{c\} \not\leq \{a, b\}$.

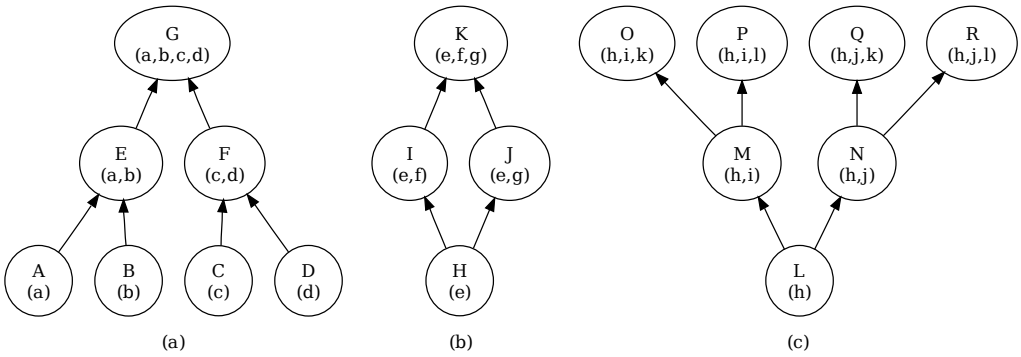



Figure 3.1: Example partial orders. Capital letters are taxon names, lowercase letters are instances of those taxa.

This definition of an `isa` hierarchy as a partially ordered set is sometimes used to define a taxonomy [Bra83, GW02, Ehr07]. It is, however, a bit too general. For example, the diagrams in Figure 3.1 are all partially ordered sets. Capital letters mark the names of the taxa, and lowercase letters mark the instances of each taxon. While (a) is certainly a taxonomy, not everyone would call (b) or (c) a taxonomy. The following definition affords more specific definitions of taxonomy.

**Definition 3.3** (Taxonomy)**.** A *taxonomy* is a 4-tuple $T = (\mathbf{N}, \leq_{isa}, \mathcal{L}_{\mathcal{T}}, \mathbf{T_C})$ consisting of a set of taxa $\mathbf{N}$, a hierarchical relation $\leq_{isa}$, a language for expressing constraints between taxa (and sets of taxa) in the taxonomy $\mathcal{L}_{\mathcal{T}}$, and a set of constraints in that language $\mathbf{T_C}$.

Note that neither this definition, nor the previous one, includes a notion of taxonomic *extent*. While the taxa in Figure 3.1 contain instances, taxonomies (and `isa` hierarchies) may be defined without reference to any instances.

The additional taxonomic constraints $\mathbf{T_C}$ are very general. Taxonomic constraints are n-ary relations on taxa. For example, a constraint might be a unary relation (*e.g.*, taxon A has at least one instance), a binary relation (*e.g.*, taxa A and B share no instances), or something more complex (*e.g.*, the extent of taxon A is the union of the extents of taxa B, C, and D).

**Example 3.4** (Example taxonomy)**.**
$T = (\mathbf{N}, \leq_{isa}, \mathcal{L}_{\mathcal{T}}, \mathbf{T_C})$
$\mathbf{N} = \{\mathbf{Animal}, \mathbf{Mammal}, \mathbf{Fish}, \mathbf{Dog}, \mathbf{Trout}\}$
$\leq_{isa}= \{(\mathbf{Mammal}, \mathbf{Animal}), (\mathbf{Fish}, \mathbf{Animal}), (\mathbf{Dog}, \mathbf{Mammal}), (\mathbf{Trout}, \mathbf{Fish})\}$
$\mathcal{L}_{\mathcal{T}}=$ set theory
$\mathbf{T_C}= \{\mathbf{Mammal} \cap \mathbf{Fish} = \emptyset\}$

Again, one can ask whether a certain interpretation $\mathcal{I}$ satisfies the taxonomy $T$. The following interpretation satisfies the hierarchical constraints of $T$ but does not satisfy the taxonomic constraint: $Animal^{\mathcal{I}} = \{a, b, c\}, Mammal^{\mathcal{I}} = \{a, b\}, Fish^{\mathcal{I}} = \{b, c\}, Dog^{\mathcal{I}} = \{a, b\}, Trout^{\mathcal{I}} = \{b, c\}$. The violation occurs because $\mathbf{Mammal} \cap \mathbf{Fish} = \{b\}$ rather than $\emptyset$.

**Definition 3.5** (Global taxonomic constraints (GTCs))**.** The difference between an `isa` hierarchy and a taxonomy is two elements ($\mathcal{L}_{\mathcal{T}}$, $\mathbf{T_C}$) that describe relationships between taxa beyond the hierarchical relation $\leq_{isa}$. The pair ($\mathcal{L}_{\mathcal{T}}$, $\mathbf{T_C}$) describes a set of additional taxonomic constraints. A *taxonomic constraint pattern* is a second-order formula that

applies to a given set of concepts. A *global taxonomic constraint* is a taxonomic constraint pattern that applies to all concepts in a given taxonomy. A *global taxonomic constraint identifier* (GTCI) is an arbitrary symbol, which, when applied to an `isa` hierarchy (or a taxonomy) via some function, generates a taxonomy.

**Example 3.6** (Example Global Taxonomic Constraint Identifier (GTCI))**.** An example GTCI might be the symbol "sibling-disjointness." This symbol is meaningless except to a function (*e.g.*, *instantiateGTCI*). For example, given an `isa` hierarchy:

$H = (\mathbf{N}, \leq_{isa})$

$\mathbf{N} = \{\mathbf{Animal}, \mathbf{Mammal}, \mathbf{Fish}, \mathbf{Dog}, \mathbf{Trout}\}$

$\leq_{isa} = \{(\mathbf{Mammal}, \mathbf{Animal}), (\mathbf{Fish}, \mathbf{Animal}), (\mathbf{Dog}, \mathbf{Mammal}), (\mathbf{Trout}, \mathbf{Fish})\}$

The function instantiateGTCI($H$,"sibling-disjointness") would create disjointness constraints between all children of every taxon that has more than one child in the $\leq_{isa}$ relation, producing a taxonomy like that described in example 3.4.

**Definition 3.7** (Articulation)**.** An articulation [MWK00, MWJ99] is a 4-tuple $A = (\mathbf{N}, \mathbf{N}, \mathcal{L}_{\mathcal{A}}, A_C)$ consisting of two sets of taxa (one from each taxonomy), a language for expressing inter-taxonomic constraints $\mathcal{L}_{\mathcal{A}}$ and an inter-taxonomic constraint $A_C$ in that language.

This definition is very general and supports many kinds of articulations. For example, one articulation might state that a taxon in $T_2$ is equivalent to the set difference of another taxon in $T_2$ and a taxon in $T_1$.

More restrictive notions of articulations will be provided in later chapters.

At times, two taxonomies might refer to taxa with the same name. In these cases, symbols representing the variables used to represent each taxonomy will be used to distinguish the two taxa by prepending the symbol representing the taxonomy, followed by a period.

**Example 3.8.** Given two taxonomies with one taxon each, that taxon symbolized by the symbol P:

$T_1 = (\{P\}, \{\}, \{\}, \{\})$
$T_2 = (\{P\}, \{\}, \{\}, \{\})$

We can refer to the $P$ in $T_1$ as $T_1.P$ and $P$ in $T_2$ as $T_2.P$. This notation will be useful when describing articulations.

**Definition 3.9** (Alignment)**.** An *alignment* consists of a pair of taxonomies and a set of articulations between them: $Align = (T_1, T_2, \mathbf{A})$. An *alignment function* between two taxonomies is a partial function which maps taxa in one taxonomy to taxa in the other, and assigns a relation to the mapping: $align : \mathcal{P}(N) \times \mathbf{T} \times \mathbf{T} \rightharpoonup \mathcal{P}(N) \times \mathcal{L}_{\mathcal{A}}$. This definition of taxonomic alignment is very similar to that of ontology alignment in [Ehr07], which in turn is based on work from [Euz04].

**Definition 3.10** (Proof line)**.** A *proof line* is a 4-tuple, $PL = (proof\_id, \varphi, J, \mathbf{P_c})$, where proof_id is a label for the proof line; $\varphi$ is a formula; $J$ is a textual *justification*, in this case a justification for that step of the proof; and $\mathbf{P_c}$ is a set of proof_ids representing the proof lines that contributed to this step of the proof.

This definition permits the creation of proof trees, which will be used to explain inferences.

**Definition 3.11** (Proof)**.** A *proof* $P = (\mathbf{PL}, \mathcal{L})$ is a pair consisting of a set of proof lines **PL** and a language $\mathcal{L}$ under which the axioms in the proof lines can be interpreted.

**Definition 3.12** (Proof Result)**.** A *proof result* $PR = (\{success, failure\}, P)$ pairs a proof with a string, either "success" or "failure." The string "success" means that $\Phi \vdash \varphi$. The string "failure" means that no proof was found either because $\Phi \nvdash \varphi$ or because the prover or calculus employed was not complete.

**Definition 3.13** (Consistency Result)**.** A *consistency result* $CR = (\{success, failure\}, \mathcal{I})$ pairs a model with a string, either "success" or "failure." The string "success" means that an interpretation $\mathcal{I}, \mathcal{I} \models \Phi$ was found. The string "failure" means that no model was found:

either because no $\mathcal{I}$ such that $\mathcal{I} \models \Phi$ could be found, or because the model finder or calculus employed was not complete.

## 3.2 Basic Operations

Defined here are basic operations used to answer the questions in section 1.4. Some of the operations require knowledge of the formal language used and the mechanism through which the calculus of that languages is applied. These parameters are designated by $\mathcal{L}$ and $\mathcal{M}$ respectively.

**Definition 3.14** (Prove)**.** Given a set of formulas $\mathbf{F}$ and a goal formula $F$, test whether the goal is implied by the formulas. Formally, does $\mathbf{F} \vdash F$.

$$prove(\mathbf{F}, F, \mathcal{L}, \mathcal{M}) \rightarrow \text{Proof Result}$$

**Definition 3.15** (Check consistency)**.** Given a list of formulas $\mathbf{F}$, check whether the formulas are logically consistent. Formally, is there an $\mathcal{I}$ such that $\mathcal{I} \models \mathbf{F}$?

$$checkConsistency(\mathbf{F}, \mathcal{L}, \mathcal{M}) \rightarrow \text{Consistency Result}$$

**Definition 3.16** (Check model)**.** Given a list of formulas $\mathbf{F}$, check whether a given interpretation, $\mathcal{I}$ is a model for those formulas. Formally, does $\mathcal{I} \models \mathbf{F}$?

$$checkModel(\mathcal{I}, \mathbf{F}, \mathcal{L}, \mathcal{M}) \rightarrow \text{Consistency Result}$$

**Definition 3.17** (Formalize Hierarchical Constraints)**.** Render a given set of hierarchical constraints $\leq_{isa}$ into a set of formulas $\mathbf{F}$ in the language $\mathcal{L}$.

$$formalizeHC(\leq_{isa}, \mathcal{L}) \rightarrow \mathbf{F}$$

**Definition 3.18** (Formalize Additional Taxonomic Constraints)**.** Render a given set of

additional taxonomic constraints, stated in a given taxonomic constraint language $\mathcal{L}_{\mathcal{T}}$, into a set of formulas $\mathbf{F}$ in the language $\mathcal{L}$.

$$formalizeATC(\mathbf{T_C}, \mathcal{L}_{\mathcal{T}}, \mathcal{L}) \rightarrow \mathbf{F}$$

**Definition 3.19** (Formalize Taxonomy)**.** Render a given taxonomy $T$ into a set of formulas $\mathbf{F}$.

$$formalizeTaxonomy(T, \mathcal{L}) \rightarrow \mathbf{F}$$

**Definition 3.20** (Formalize Articulations)**.** Render a set of articulations $\mathbf{A}$ into a set of formulas $\mathbf{F}$.

$$formalizeArticulations(\mathbf{A}, \mathcal{L}) \rightarrow \mathbf{F}$$

**Definition 3.21** (Instantiate Global Taxonomic Constraint Identifier)**.** Given an `isa` hierarchy $H$ or a taxonomy $T$ and an global taxonomic constraint identifier $GTCI$, apply the GTCI and return the resulting taxonomy.

$$instantiateGTCI(H, GTCI) \rightarrow T$$

$$instantiateGTCI(T, GTCI) \rightarrow T$$

**Definition 3.22** (Calculate All Possible Hierarchical Constraints)**.** Given a taxonomy $T$ or an `isa` hierarchy $H$, return a set of all possible sets of hierarchical constraints.

$$calculateAllHC(T) \rightarrow \mathcal{P}(N \times N)$$

$$calculateAllHC(H) \rightarrow \mathcal{P}(N \times N)$$

This operation returns all possible sets of hierarchical constraints for a taxonomy or `isa`

hierarchy, taking into account only the taxa in $\mathbf{N}$.

**Example 3.23** (Example application of calculateAllHC operation)**.**

$H = (\mathbf{N}, \leq_{isa})$

$\mathbf{N} = \{\mathbf{A}, \mathbf{B}\}$

$\leq_{isa} = \{(\mathbf{B}, \mathbf{A})\}$

$calculateAllHC(H) \rightarrow \{$

$\{\},$

$\{(\mathbf{A}, \mathbf{A})\}, \{(\mathbf{A}, \mathbf{B})\}, \{(\mathbf{B}, \mathbf{A})\}, \{(\mathbf{B}, \mathbf{B})\},$

$\{(\mathbf{A}, \mathbf{A}), (\mathbf{A}, \mathbf{B})\}, \{(\mathbf{A}, \mathbf{A}), (\mathbf{B}, \mathbf{A})\}, \{(\mathbf{A}, \mathbf{A}), (\mathbf{B}, \mathbf{B})\},$

$\{(\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A})\}, \{(\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{B})\}, \{(\mathbf{B}, \mathbf{A}), (\mathbf{B}, \mathbf{B})\},$

$\{(\mathbf{A}, \mathbf{A}), (\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A})\}, \{(\mathbf{A}, \mathbf{A}), (\mathbf{B}, \mathbf{A}), (\mathbf{B}, \mathbf{B})\},$

$\{(\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A}), (\mathbf{B}, \mathbf{B})\}, \{(\mathbf{A}, \mathbf{A}), (\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A}), (\mathbf{B}, \mathbf{B})\}$

$\}$

Note that because $\leq_{isa}$ is a partial order, some of the sets in the power set constrain some of the taxa to be equivalent. For example, the set of constraints: $\{(\mathbf{A}, \mathbf{B}), (\mathbf{B}, \mathbf{A})\}$ can only be a partial order if $A = B$. This may contradict constraints in $\mathbf{T_C}$, or it may be incompatible with a given interpretation $\mathcal{I}$.

**Definition 3.24** (Calculate All Possible Additional Taxonomic Constraints)**.** Given a taxonomy $T$ or an `isa` hierarchy $H$, return a set of all possible (both satisfiable and not) additional taxonomic constraints.

$$calculateAllATC(T) \rightarrow \mathbf{T_C}$$

$$calculateAllATC(H, \mathcal{L}_{\mathcal{T}}) \rightarrow \mathbf{T_C}$$

This operation returns all well-formed formulas for a given taxonomy according to the language $\mathcal{L}_{\mathcal{T}}$ without regard to whether or not any given formula is satisfiable. Unlike the set of hierarchical constraints derived using *calculateAllHC*, the set of possible additional

taxonomic constraints may potentially be infinite, and therefore incalculable. For example, if the language for expressing ATCs ($\mathcal{L}_\mathcal{T}$) is full first-order logic, then there are infinite ways of creating formulas using the taxa in $T$ (*e.g.*, $A \rightarrow A$, $A \rightarrow A \wedge A$, $A \rightarrow A \wedge A \wedge A$ ...). Throughout this dissertation, several different examples of $\mathcal{L}_\mathcal{T}$ will be presented, and in each, the set of possible well-formed formulas using a finite set of taxa will be finite.

**Definition 3.25** (Calculate All Possible Taxonomies)**.** Given a set of taxa **N**, an ordering relation $\leq_{isa}$, and a language for stating taxonomic constraints $\mathcal{L}_\mathcal{T}$, return a set of all possible taxonomies.

$$calculateAllTaxonomies(\mathbf{N}, \leq_{isa}, \mathcal{L}_\mathcal{T}) \rightarrow \mathbf{T}$$

**Definition 3.26** (Calculate All Possible Articulations)**.** Given a pair of taxonomies and an articulation language $\mathcal{L}_\mathcal{A}$ return the set of all possible articulations, without regard to whether any given articulation is satisfiable.

$$calculateAllArticulations(T, T, \mathcal{L}_\mathcal{A}) \rightarrow \mathbf{A}$$

This operation returns all well-formed formulas for articulations, given two taxonomies and a language for expressing articulations. As was the case for *calculateAllATC*, this may be incalculable if the language $\mathcal{L}_\mathcal{A}$ permits the construction of infinite well-formed formulas given two taxonomies.

**Definition 3.27** (Calculate All Possible Alignments)**.** Given an alignment, or a pair of taxonomies and an articulation language $\mathcal{L}_\mathcal{A}$, return a set of all possible alignments.

$$calculateAllAlignments(Align) \rightarrow \mathbf{Align}$$

$$calculateAllAlignments(T, T, \mathcal{L}_\mathcal{A}) \rightarrow \mathbf{Align}$$

## 3.3 Formalized Questions

We can now formalize some of the questions in section 1.4. In this section, lists will be marked using square brackets (*e.g.*, a list of proof results would be represented as [Proof Result]). The formalization of questions not provided here awaits further work.

**Question 1.3 Representation.** Taxonomies and articulations between them have been formalized above. In later chapters, the formalizations will be further grounded in specific languages.

**Question 1.4 Uncertainty.** Definitions for relations between taxa occur in several places in the above formalization: the hierarchical relation $\leq_{isa}$ used in an `isa` hierarchy and a taxonomy, intra-taxonomic relations in $\mathcal{L}_{\mathcal{T}}$, and articulation relations in $\mathcal{L}_{\mathcal{A}}$. Uncertainty can be represented in these languages by using *disjunctive relations*. For example, if a language $\mathcal{L}_{\mathcal{T}}$ contains two relations, "equals" and "is included in," the subset relation $\subseteq$ would be represented by a disjunction: "equals or is included in."

**Question 1.5 Consistency of taxonomies.** Given a formal language $\mathcal{L}$, and a model finder $\mathcal{M}$ the consistency of a given taxonomy $T$ may be decided as follows:

$$taxonomyConsistent \quad : \quad checkConsistency( \\ formalizeTaxonomy(T, \mathcal{L}), \mathcal{L}, \mathcal{M}) \rightarrow \text{Consistency Result}$$

**Question 1.6 Models.** Given an interpretation $\mathcal{I}$, a formal language $\mathcal{L}$, and a model finder $\mathcal{M}$, we can ask:

$$interpModelsTax \quad : \quad checkModel( \\ \mathcal{I}, formalizeTaxonomy(T, \mathcal{L}), \mathcal{L}, \mathcal{M}) \rightarrow \text{Consistency Result}$$

**Question 1.7 Consistency of alignment.** Given a formal language $\mathcal{L}$, and a model finder $\mathcal{M}$ the consistency of a given alignment $align = (T_1, T_2, \mathbf{A})$ may be decided as follows:

$$
\begin{aligned}
consistentAlignment \quad : \quad & checkConsistency( \\
& formalizeTaxonomy(T_1, \mathcal{L}) \ \cup \\
& formalizeTaxonomy(T_2, \mathcal{L}) \ \cup \\
& formalizeArticulations(\mathbf{A}, \mathcal{L}), \mathcal{L}, \mathcal{M}) \rightarrow \\
& \text{Consistency Result}
\end{aligned}
$$

**Question 1.8(i) Deductive closure for a taxonomy.** Given a consistent taxonomy, a formal language $\mathcal{L}$ and a theorem prover $\mathcal{M}$

$$
\begin{aligned}
taxonomyClosure \quad : \quad & [\text{Proof Result} \mid \\
& prove(formalizeTaxonomy(T, \mathcal{L}), \\
& formalizeITC(itc, \mathcal{L}), \mathcal{L}, \mathcal{M}), \\
& itc \in \{calculateAllHC(T) \cup calculateAllATC(T)\}]
\end{aligned}
$$

This definition returns a list of proof results, with some proofs being successful while others fail. This list can then be filtered for just the successful proofs to provide a list of the relations that hold between taxa in a taxonomy.

**Question 1.8(ii) Deductive closure for articulations.** Given two consistent taxonomies $T_1$, $T_2$, an alignment $Align = (T_1, T_2, \mathbf{A})$, a formal language $\mathcal{L}$, and a theorem prover $\mathcal{M}$

$$articulationClosure \ : \ [\text{Proof Result} \ | \ prove(formalizeArticulations(\mathbf{A}, \mathcal{L}), formalizeArticulations([art], \mathcal{L}), \mathcal{L}, \mathcal{M}), art \in calculateAllArticulations(T_1, T_2, \mathcal{L}_{\mathcal{A}})]$$

This definition returns a list of proof results, with some proofs being successful while others fail. This list can then be filtered for just the successful proofs to provide a list of articulations that hold between two taxonomies.

**Question 1.8(iii) Deductive closure for an alignment.** Given an alignment $Align = (T_1, T_2, \mathbf{A})$, a formal language $\mathcal{L}$, and a theorem prover $\mathcal{M}$

$$\begin{aligned} alignmentClosure \ = \ & taxonomyClosure(T_1, \mathcal{L}, \mathcal{M}) \cup \\ & taxonomyClosure(T_2, \mathcal{L}, \mathcal{M}) \cup \\ & articulationClosure(\mathbf{A}, \mathcal{L}, \mathcal{M}) \end{aligned}$$

**Question 1.12(i) Minimal Inter-Taxonomic Constraints.** Given a taxonomy $T$ return a set of taxonomies $T_1, T_2, \ldots, T_n$ where each taxonomy $T_n$ has the same deductive closure as $T$, and the size of the set of constraints $\leq_{isa} \cup \mathbf{T_C}$ is smaller for $T_n$ than for $T$. This list can be further processed to retrieve the taxonomies with the smallest number of constraints.

$$
\begin{aligned}
minimalITCs \;:\; & [T' \mid \\
& taxonomyClosure(T', \mathcal{L}, \mathcal{M}) = taxonomyClosure(T, \mathcal{L}, \mathcal{M}), \\
& T' \in calculateAllTaxonomies(\mathbf{N}, \leq_{isa}, \mathcal{L}_{\mathcal{T}})), \\
& |\leq'_{isa_T} \cup \mathbf{T_{C_T}}'| < |\leq_{isa_T} \cup \mathbf{T_{C_T}}|]
\end{aligned}
$$

The vertical bars || signify the cardinality of the set between them.

**Example 3.28** (Two minimal sets of formulas)**.** A taxonomy formalized into first-order logic with the following set of formulas: $\{\forall x : B(x) \rightarrow A(x), \forall x : C(x) \rightarrow A(x), \forall x : B(x) \wedge C(x), \forall x : \neg(\neg B(x) \vee \neg C(x))\}$ has two sets of minimal formulas: $\{\forall x : B(x) \rightarrow A(x), \forall x : C(x) \rightarrow A(x), \forall x : B(x) \wedge C(x)\}$ and $\{\forall x : B(x) \rightarrow A(x), \forall x : C(x) \rightarrow A(x), \forall x : \neg(\neg B(x) \vee \neg C(x))\}$.

**Question 1.12(ii) Minimal Articulations.** Given an alignment $Align = (T_1, T_2, \mathbf{A})$ return a set of alignments $Align_1, Align_2, \ldots, Align_n$ where alignment $Align_n$ has the same deductive closure as $Align$, and the size of the set of articulations $\mathbf{A}$ is smaller in $Align_n$ than in $Align$.

$$
\begin{aligned}
minimalArticulations \;:\; & [Align' \mid \\
& alignmentClosure(Align', \mathcal{L}, \mathcal{M}) = \\
& alignmentClosure(Align, \mathcal{L}, \mathcal{M}), \\
& Align' \in calculateAllAlignments(Align)), \\
& |A'| < |A|]
\end{aligned}
$$

## 3.4 Contributions and Future Work

This chapter formalizes taxonomies, articulations, and alignments. It also gives preliminary definitions of terms related to logical proofs. This latter set of terms will probably be revised once work on explanations has been undertaken more fully. In addition to providing formal definitions for the major terms used in this dissertation, the chapter defines key operations used by the CleanTax approach. These operations are then used to formalize some of the questions described in the dissertation's introduction. Not all of the questions in section 1.4 have been addressed in this section. Other questions will be formalized in later chapters, and others remain for future work.

# Chapter 4

# Taxonomy Alignment

## 4.1 Overview and Objectives

This chapter[1] concretizes the definitions of taxonomies, alignments, uncertainty, consistency, and inference given in Chapter 3 using monadic first-order logic.

### 4.1.1 Prior Work

#### In Computer Science

The process of finding alignments (often called *matching*) has been studied in many contexts, for example, database schema matching [RB01], XML matching [MHH+01], and ontology mapping in the semantic web [NM03, KS03, ST05, SSW05, SE07]. This research spans a vast landscape of techniques and scenarios and a number of extensive reviews have already done an excellent job of capturing the multitude of alignment systems [KS03, Euz04, Ehr07, SE07]. Rather than recapitulate this work, the following focuses on systems similar to that in this thesis, notably systems working in situations where the instances and character-based definitions of taxa are not known and the names of taxa are ignored. This approach is not meant to exclude the other techniques, such as those that

[1] Much of this chapter is drawn from [TL07] and [Tha08].

do a lexical analysis of taxon names, or those that use instances to guide alignment. These methods are all complementary.

A key differentiator between solutions to the alignment problem involves the languages used to express the data being aligned (XML schemata, ontologies, taxonomies, *etc.*), and the language of the articulations involved in that alignment. Most researchers have utilized some decidable subset of first-order logic, such as description logics or propositional logic. These logics will be described in more detail in Chapter 5. One exception to this rule is the OntoMerge [DMQ05] system. OntoMerge represents ontologies and bridging rules between them in a strongly-typed full first-order language called Web-PDDL [MD02], which is an extension of the Planning Domain Definition Language (PDDL) [McD98]. OntoMerge is designed to merge ontologies, translate data represented in one ontology into a second ontology, and query across ontologies. Although the taxonomies and articulations described here could be represented in the Web-PDDL language, OntoMerge explicitly ignores the alignment aspect of the ontology translation process, assuming that bridging axioms are provided by experts or other systems. This alternative focus reduces the relevance of OntoMerge to the majority of this thesis. However, it will be revisited in Chapter 6, where taxonomy merging is discussed. The systems closest to the approach described in this dissertation (hereafter called the CleanTax approach) are CtxMatch [BMSZ03] and its enhanced version S-Match [GSY04] from the University of Trento. Both of these systems take as input two tree-like structures (XML schemas or ontologies) and apply a number of matching techniques to generate an alignment. One of these techniques casts the input into propositional logic formulas and computes consistency and new logical relationships using a SAT reasoner. The main differences between the CtxMatch and S-Match approach and the CleanTax approach involve the actual logical representations used. CtxMatch and S-Match represent articulations as logical equivalence, subsumption, or intersection. As will be seen, CleanTax supports a richer vocabulary for representing articulations —one that in some cases supports more tractable reasoning than that available to CtxMatch and S-Match (see Chapter 5 for more on this).

Another significant difference between the current work and other approaches, including CtxMatch and S-Match, is the way the CleanTax approach represents and processes uncertainty. In CleanTax, uncertain articulations are represented with disjunctive relationships. In other approaches, uncertainty is typically represented by attaching probabilities to articulations [PTU03, Ehr07]. Using disjunctive relationships to represent uncertainty affords some interesting strategies for minimizing uncertainty (see section 9.4).

### In Life Sciences

Taxonomies have been studied outside the realm of knowledge representation - primarily in the life sciences. For example, in 2005, the Taxonomic Data Working Group (TDWG) [tdw], an international not-for-profit organization that develops standards for biodiversity data, ratified the Taxonomic Concept Schema (TCS) [TCS]. TCS is an XML Schema that defines a syntax for describing *taxonomic concepts*. The TCS includes a list of terms that may be used to define the relationships between two different taxonomic concepts. This list includes set-theoretic terms, such as *is congruent to* and *excludes*. However, some of the terms are not well defined. For example, it is unclear whether *is included in* means *proper subset* ($\subsetneq$), or if it means *subset* ($\subseteq$), which includes the possibility that both sets are equal. It also includes more vague relationships such as *has synonym*. These vague relationships are needed in TCS, as it aims to provide a standard for information providers to communicate information about their data. If an information provider wants to use a mathematically imprecise relationship such as *has synonym*, TCS must support that relationship. However, unless the meaning of such relations is specified more precisely, its utility for automated reasoning will be diminished.

Beach et al. [BPB93] introduced, and Berendsohn [Ber95] elaborated, the notion of a *potential taxon*, which identifies a taxonomic concept by referencing the context in which the name is used; *e.g.*, *Hypnum flagellare* Dicks. sec. Mönkemeyer 1927. This notion is central to the MoReTaX project [Ber03], in which potential taxa are considered sets of objects, and the relationships between them are described in precise set-theoretic terms. The five

so-called *basic relations* that may hold between any two potential taxa (or, in fact, any two non-empty sets) $A$ and $B$ are: (i) congruence ($A \equiv B$), (ii) proper inclusion ($A \subsetneq B$), (iii) proper inverse inclusion ($A \supsetneq B$), (iv) partial overlap ($A \oplus B$), and (v) exclusion (disjointness) ($A \,!\, B$). Geoffroy and Güntsch [GG03] study the problem of *propagating* knowledge about such binary relationships between taxa: *e.g.*, what can we say about the relationship between potential taxa $A$ and $C$, provided we only know that $A \supsetneq B$ and $B \oplus C$? Inspection of all possibilities allows one to deduce that $A \supsetneq C$ or $A \oplus C$, but none of the other three options $\equiv$, $\subsetneq$, or ! is possible between $A$ and $C$. Thus, the authors study *combined relationships* (*i.e.*, disjunctions of basic relations: *e.g.*, $\{\supsetneq, \oplus\}$) and demonstrate how these may be composed to propagate taxonomic knowledge in a *potential taxon graph*. Rules for knowledge propagation in a taxon graph are given as `if-then` rules, embedded in the MoReTaX system; thus, computing with taxon relations is handled programmatically. In a sense, the work presented here is a formal grounding and extension of the MoReTaX program.

### 4.1.2 Goals and Outcomes

This chapter provides a novel formalization of taxonomies and taxonomic alignment using monadic first-order logic. The formalization supports questions about taxonomic consistency, uncertainty in articulations, and alignment consistency. The formalization also supports the inference of unstated intra-taxonomic relationships and articulations. In order to achieve these goals, the formalization is implemented in a software framework, which will be described in more detail in Chapter 8. This framework is applied to real-world datasets to demonstrate the detection of inconsistencies in taxonomies and articulations, and the inference of new relations between taxa.

## 4.2 Monadic First-Order Logic (MFOL)

The notion of a formal language was introduced in Chapter 2. One of the most expressive formal languages is first-order logic, $\mathcal{L}_{\text{FOL}}$. Although first-order logic is both sound and complete, it is also undecidable; there is no method for showing that any arbitrary sentence in $\mathcal{L}_{\text{FOL}}$ is provable within $\mathcal{L}_{\text{FOL}}$ [Chu36, Tur36]. However, a subset of first-order logic, called monadic first-order logic ($\mathcal{L}_{\text{MFOL}}$), is sound, complete, and decidable [Beh22], and this subset is adequate for reasoning about set constraints [BGW93]. The key differences between full first-order logic and monadic first-order logic are that in monadic first-order logic, all predicates take a single argument (as opposed to permitting predicates with arbitrary arity), and monadic first-order logic has no function symbols. The formal syntax and semantics of both monadic first-order logic and full first-order logic are presented in Appendix sections A.1.2 and A.1.1 respectively. In this chapter, taxonomies and articulations between them will be formalized and translated into $\mathcal{L}_{\text{MFOL}}$.

## 4.3 Formalizing Taxonomies as Monadic First-Order Logic Constraints

As defined in Chapter 3, a taxonomy is a 4-tuple $T = (\mathbf{N}, \leq_{isa}, \mathcal{L}_{\mathcal{T}}, \mathbf{T_C})$ consisting of a set of taxa $\mathbf{N}$, a hierarchical relation $\leq_{isa}$, a language for expressing constraints between taxa (and sets of taxa) in the taxonomy $\mathcal{L}_{\mathcal{T}}$, and a set of constraints in that language $\mathbf{T_C}$. To ground this definition in $\mathcal{L}_{\text{MFOL}}$, we need to specify three things: (i) how to formalize the $\leq_{isa}$ relation, (ii) what the constraint language $\mathcal{L}_{\mathcal{T}}$ is, and (iii) how to translate constraints in $\mathbf{T_C}$ into monadic first-order logic constraints.

### 4.3.1 Formalizing Hierarchical Constraints ($\leq_{isa}$)

The relation $N \leq_{isa} M$ is formalized in $\mathcal{L}_{\text{MFOL}}$ as follows: $\forall x$: $N(x) \rightarrow M(x)$, stating that if $x$ is in $N$, then $x$ is also in $M$. Note that $N$ and $M$ are unary predicates, in keeping with

the restrictions of monadic logic. An interpretation $\mathcal{I}$ satisfies this constraint iff $N^{\mathcal{I}} \subseteq M^{\mathcal{I}}$. With this logic formalization, the containment relation defined by a pair of taxa ($N$,$M$) occuring in $\leq_{isa}$ is true if either $N^{\mathcal{I}} \subsetneq M^{\mathcal{I}}$ (proper containment) or $N^{\mathcal{I}} = M^{\mathcal{I}}$ (set equality).

In this way, we can associate with each taxonomy $T$ a set of $\mathcal{L}_{\text{MFOL}}$ constraints $\Phi_T^{\mathtt{isa}}$ that capture the meaning of the hierarchical constraints:

$$\Phi_T^{\mathtt{isa}} := \{\ \forall x\colon N(x) \rightarrow M(x)\ \mid\ (N, M) \in \leq_{isa}\ \}$$

Once we have formalized a taxonomy $T$ as a set of logical constraints $\Phi_T^{\mathtt{isa}}$, we can ask whether or not a given labeling $\mathcal{I}$ of elements agrees with the taxonomy. Formally: does $\mathcal{I} \models \Phi_T^{\mathtt{isa}}$ hold?

**Example 4.1.** Consider the following oversimplified taxonomy, created by a taxonomist named Carl:

$$T_{\text{Carl}} = (\ \underbrace{\{\mathtt{Fox}, \mathtt{Dog}, \mathtt{Canis}\}}_{\mathbf{N}},\ \underbrace{\{(\mathtt{Fox}, \mathtt{Canis}),\ (\mathtt{Dog}, \mathtt{Canis})\}}_{\leq_{isa}}\ )$$

Carl's taxonomy $T_{\text{Carl}}$, interpreted as a set of $\mathcal{L}_{\text{MFOL}}$ formulas $\Phi_{T_{\text{Carl}}}^{\mathtt{isa}}$ constrains the interpretation of the taxa $\mathbf{N} = \{\mathtt{Fox}, \mathtt{Dogs}, \mathtt{Canis}\}$. Now consider a museum collection organized by a curator named Ed who has labeled the specimens in his collection with taxonomic names. We can view this labeling as a logic interpretation $\mathcal{I}_{\text{Ed}}$ that assigns to each taxon name $N \in \mathbf{N}$ a subset $N^{\mathcal{I}_{\text{Ed}}} \subseteq D$ of elements from the underlying domain (here: specimens). Let Ed's specimen collection include $\{\mathtt{s}_1, \mathtt{s}_2\}$, and let his interpretion of Carl's names be $\mathtt{Fox}^{\mathcal{I}_{\text{Ed}}} = \{\mathtt{s}_1\}$, $\mathtt{Vulpes}^{\mathcal{I}_{\text{Ed}}} = \{\mathtt{s}_1\}$, $\mathtt{Dog}^{\mathcal{I}_{\text{Ed}}} = \{\mathtt{s}_2\}$, and $\mathtt{Canis}^{\mathcal{I}_{\text{Ed}}} = \{\mathtt{s}_2\}$.

We can now ask whether Ed's specimen labeling $\mathcal{I}_{\text{Ed}}$ *satisfies* the constraints imposed by Carl's taxonomy $T_{\text{Carl}}$, or more formally: does $\mathcal{I}_{\text{Ed}} \models \Phi_{T_{\text{Carl}}}^{\mathtt{isa}}$ hold?

Given Carl's taxonomy and Ed's interpretation, the answer is no. Carl requires $\forall x\colon \mathtt{Fox}(x) \rightarrow \mathtt{Canis}(x)$, but for Ed's labeling $\mathcal{I}_{\text{Ed}}$ we have $\mathtt{Fox}^{\mathcal{I}_{\text{Ed}}} \nsubseteq \mathtt{Canis}^{\mathcal{I}_{\text{Ed}}}$. Therefore $\mathcal{I}_{\text{Ed}} \not\models \Phi_{T_{\text{Carl}}}^{\mathtt{isa}}$, *i.e.*, Ed's specimen labeling is *not* a model of (or: does not satisfy) Carl's

taxonomy constraints.

Assume, as in Example 4.1, that an interpretation $\mathcal{I}$ is given and that we wish to check whether $\mathcal{I}$ is a model of (*i.e.*, satisfies) $\Phi_T$, where $\Phi_T$ captures constraints about the taxa in $T$. Model-checking whether $\mathcal{I} \models \Phi_T$ holds can be done efficiently even for large interpretations (here: labeled specimen collections) $\mathcal{I}$, as it precisely corresponds to evaluating the constraints $\Phi_T$ against a given database instance $\mathcal{I}$. Thus, for each constraint $\varphi \in \Phi_T$ we can simply run a yes/no (*i.e.*, boolean) SQL query $Q_\varphi$ against a relational database instance $\mathcal{I}$. Here, $\Phi_T$ is not limited to `isa` constraints, but can include *any* finite set of monadic first-order constraints – a much more expressive language than $\Phi_T^{\texttt{isa}}$.

While model-checking $\mathcal{I} \models \Phi_T$ does not require a full-fledged automated first-order reasoner (SQL query evaluation is sufficient), we will encounter harder *implication problems*, where no interpretation $\mathcal{I}$ is given. Instead we wish to know whether a "piece of taxonomic knowledge" $\varphi$ follows from a set of taxonomy constraints $\Phi_T$, *independent* of $\mathcal{I}$. Determining this, *i.e.*, whether $\Phi_T \models \varphi$ holds, requires $\mathcal{L}_{\text{MFOL}}$ reasoning.

### 4.3.2 Formalizing $\mathcal{L}_\mathcal{T}$: The Language of Taxonomic Constraints

$\mathcal{L}_\mathcal{T}$ is the language used to describe constraints between taxa in a taxonomy. $\mathcal{L}_\mathcal{T}$ could be full first-order logic. However, reasoning in $\mathcal{L}_{\text{FOL}}$ is undecidable, rendering it an unsatisfactory choice unless the power of full $\mathcal{L}_{\text{FOL}}$ is necessary. However, $\mathcal{L}_{\text{FOL}}$ is more expressive than needed for the questions asked in this thesis, and we can shift to the less expressive monadic first-order logic.

We have already seen how hierarchicial constraints, initially represented as partial order relations $\leq_{isa}$ are rendered into $\mathcal{L}_{\text{MFOL}}$. A similar translation process occurs with the constraints represented in $\mathcal{L}_\mathcal{T}$. A simple strategy would be to make $\mathcal{L}_\mathcal{T}$ exactly $\mathcal{L}_{\text{MFOL}}$, eliminating the need to translate from $\mathcal{L}_\mathcal{T}$ to $\mathcal{L}_{\text{MFOL}}$. However, part of the reason for creating a taxonomic constraint language is to provide a natural set of relations for modelers to apply within a given context. For example, in some domains, it may be natural to permit


(i) *congruence* (ii) *proper inclusion* (iii) *proper inverse inclusion* (iv) *partial overlap* (v) *exclusion*

Figure 4.1: The 5 basic relations: (i) $N \equiv M$, (ii) $N \subsetneq M$, (iii) $N \supsetneq M$, (iv) $N \oplus M$, and (v) $N \,!\, M$

only pairwise constraints between taxa. In other domains more complex constraints may apply. This chapter and the ones to follow, will describe a number of such sets of constraints, defining various versions of $\mathcal{L}_{\mathcal{T}}$.

Let us turn, then, to an example of a $\mathcal{L}_{\mathcal{T}}$ called $\mathcal{L}_{\mathcal{T}_1}$, one suited for application to systematic biology. $\mathcal{L}_{\mathcal{T}_1}$ supports several different types of constraints, which are divided here into three categories: (i) simple relations between taxa, (ii) disjunctive relations between taxa, (iii) and other taxonomic constraints. The constraint types described below have been chosen to match relationships described in the TDWG [tdw] standard for describing biological taxonomies [TCS], as well as to include other constraint types that hold in various settings in systematic biology. Investigations into the relevance of $\mathcal{L}_{\mathcal{T}_1}$ for other domains remains for future work.

### Basic Relations Between Taxa

The simplest constraints in $\mathcal{L}_{\mathcal{T}_1}$ occur between two taxa and contain no uncertainty. Let $N, M$ denote two non-empty sets.[2] Then exactly one of the following five *basic relations* must hold between them (*cf.* Figure 4.1): (i) congruence (equivalence) ($N \equiv M$), (ii) proper inclusion ($N \subsetneq M$), (iii) proper inverse inclusion ($N \supsetneq M$), (iv) partial overlap ($N \oplus M$), and (v) exclusion (disjointness) ($N \,!\, M$).

Let $\mathbb{B}_5 = \{\equiv, \subsetneq, \supsetneq, \oplus, !\}$ denote this set of basic relations. The importance of the $\mathbb{B}_5$ relations (and their combinations) for capturing and reasoning with constraints between (potential) taxa has been noted before, both in computer science [RCC92, Ben94, JD97]

[2]Thus, the extents of $N$ and $M$ have been fixed as $N^I$ and $M^I$ via an (otherwise unimportant) interpretation $\mathcal{I}$. Non-empty sets are critical to maintain the mutual exclusivity of the $\mathbb{B}_5$ relations. If empty sets were permitted, an empty set would be simultaneously disjoint from, and included in all non-empty sets.

and in biology [GB03a, TCS, FPW07].

These five basic relations can be translated into $\mathcal{L}_{\text{MFOL}}$ as follows:

- **Congruence:** for each $N \equiv M$ constraint in $\mathbf{T_C}$, add the formula:

  $$\forall x\colon N(x) \leftrightarrow M(x)$$

- **Proper inclusion:** for each $N \subsetneq M$ constraint in $\mathbf{T_C}$, add the formula:

  $$\forall x\colon N(x) \rightarrow M(x) \wedge \exists a\colon M(a) \wedge \neg N(a)$$

- **Proper inverse inclusion:** for each $N \supsetneq M$ constraint in $\mathbf{T_C}$, add the formula:

  $$\forall x\colon M(x) \rightarrow N(x) \wedge \exists a\colon N(a) \wedge \neg M(a)$$

- **Partial overlap:** for each $N \oplus M$ constraint in $\mathbf{T_C}$, add the formula:

  $$\exists a \exists b \exists c\colon N(a) \wedge M(a) \wedge N(b) \wedge \neg M(b) \wedge \neg N(c) \wedge M(c)$$

- **Exclusion** for each $N \,!\, M$ constraint in $\mathbf{T_C}$, add the formula:[3]

  $$\neg \exists x\colon N(x) \wedge M(x)$$

Unlike $\Phi_T^{\mathtt{isa}}$, the constraints $\Phi_T^{\mathbb{B}_5}$ for a $\mathbb{B}_5$-taxonomy graph $T$ may be inconsistent. For example, consider $\mathbf{T_C} = \{B \equiv A,\ C \subsetneq A,\ B \,!\, C\}$. The resulting set of constraints $\Phi_T^{\mathbb{B}_5}$ is inconsistent: Since $B \equiv A$ it follows that $C \subsetneq B$, contradicting $B \,!\, C$.

### Disjunctive Relations Between Taxa

As noted, exactly one of the five basic relations $\mathbb{B}_5$ must hold between any two sets $N^I$ and $M^I$, *i.e.*, provided an interpretation $\mathcal{I} = (D, I)$ is given for all taxa $N$, $M$ under consideration. However, this is usually not the case — one often does not have complete knowledge corresponding to an interpretation $\mathcal{I}$ of taxa. Instead, one must reason with *partial* taxonomic knowledge given in the form of constraints that are to be satisfied by *any*

[3] Two equivalent, alternative formalizations are: (i) $\forall x\colon \neg N(x) \vee \neg M(x)$ and (ii) $\forall x\colon N(x) \rightarrow \neg M(x)$.

interpretation $\mathcal{I}$. This means $\mathcal{L}_{\mathcal{T}_1}$ should be able to express not only the basic relations $\mathbb{B}_5$, but the larger set of possible combinations between them. By disjunctively combining the basic relations $\mathbb{B}_5$ in all possible ways, we obtain 32 ($=2^5$) relations, one for each subset of $\mathbb{B}_5$; we denote this set of *disjunctive relations* by $\mathbb{R}_{32}$. For example, $\mathbb{R}_{32}$ contains $\{\equiv, \subsetneq\}$ which is read as the disjunction $(N \equiv M) \vee (N \subsetneq M)$. This is equivalent to a standard `isa` relation, *i.e.*, $N \subseteq M$. $\mathbb{R}_{32}$ also contains extreme cases of relations, *e.g.*, $\{\equiv, \subsetneq, \supsetneq, \oplus, !\}$, the disjunction of all basic relations, stating that nothing about the relationship between two taxa is known.

Each of these $\mathbb{R}_{32}$ can be reduced to $\mathcal{L}_{\text{MFOL}}$ with a set of constraints $\Phi_T^{\mathbb{R}_{32}}$ by simply taking the disjunction of the translated constituents of the disjunctive relation. Once we have done so, we can test whether some taxonomic knowledge $\varphi$ is implied by $T$ by checking whether $\Phi_T^{\mathbb{R}_{32}} \models \varphi$ holds. Similarly, we can test whether $\Phi_T^{\mathbb{R}_{32}}$ is consistent. Just as $\Phi_T^{\mathbb{B}_5}$ above, $\Phi_T^{\mathbb{R}_{32}}$ is a subset of $\mathcal{L}_{\text{MFOL}}$, so implication and satisfiability are decidable.

**Corollary 4.2.** The implication problem for taxonomies (formalized as constraints $\Phi_T \subseteq \mathcal{L}_{\text{MFOL}}$) is decidable.

Let $\Phi_T \subseteq \mathcal{L}_{\text{MFOL}}$ be a finite set of $\mathcal{L}_{\text{MFOL}}$ constraints formalizing a taxonomy $T$. Then $\Phi_T \models \varphi$ is decidable. Since there are sound and complete first-order calculi (implemented by automated first-order reasoners), we can check whether $\varphi$ can be *proven* from $\Phi_T$ using an automated reasoner, denoted $\Phi_T \vdash \varphi$. Similarly, the corollary implies that there is an algorithm to check whether a set of taxonomy constraints $\Phi_T$ is *consistent*. However, automated reasoning over large sets of taxonomy constraints may still be infeasible in practice.

**Example 4.3.** Translating a disjunctive relation into $\mathcal{L}_{\text{MFOL}}$. The disjunctive relation $N\{\equiv, \subsetneq\}M$ translates to the following in $\mathcal{L}_{\text{MFOL}}$:

$(\forall x\colon N(x) \leftrightarrow M(x)) \vee (\forall x\colon N(x) \rightarrow M(x) \wedge \exists a\colon M(a) \wedge \neg N(a))$

This is simply the disjunction of the translations of $(N \equiv M)$ and $(N \subsetneq M)$.

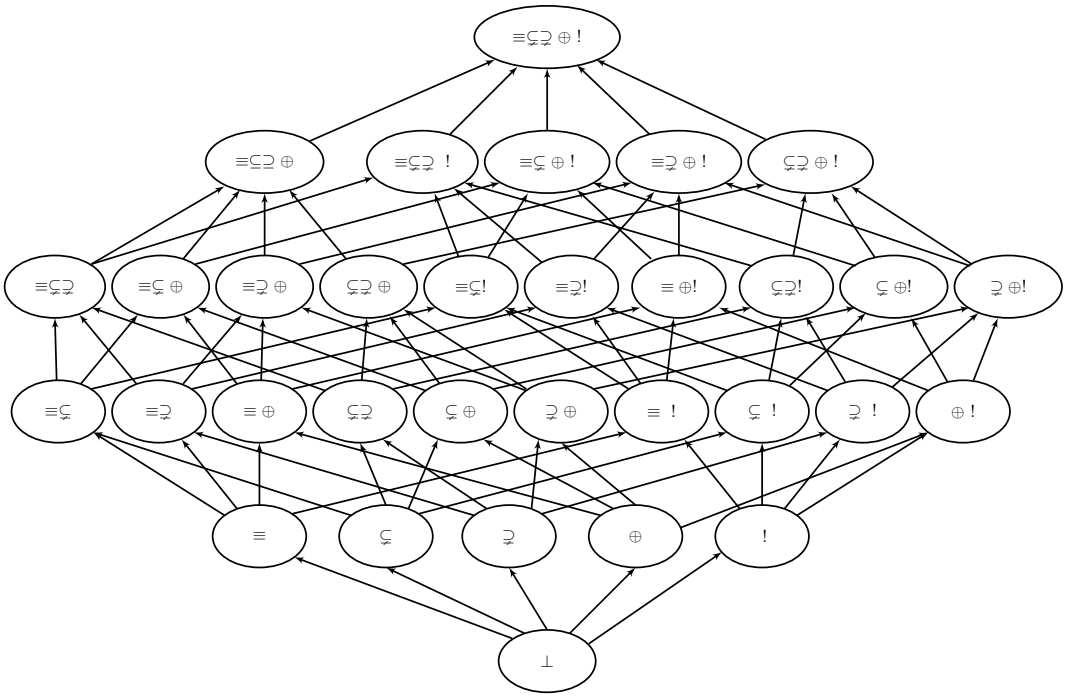



Figure 4.2: The $\mathbb{R}_{32}$ lattice. Each node represents one of the $\mathbb{R}_{32}$ relations. The power set of $\mathbb{B}_5$ induces the complete lattice shown here. The top element $\{\equiv,\subsetneq,\supsetneq,\oplus,!\}$ represents a complete lack of knowledge about the relationship between two taxa. Each layer-1 node represents one of the $\mathbb{B}_5$. Nodes in between represent some level of uncertainty about the relationship between two taxa.

**The $\mathbb{R}_{32}$ lattice**. The constraints in $\mathbb{R}_{32}$ form the lattice shown in Figure 4.2. The top of the lattice $\top = \{\equiv, \subsetneq, \supsetneq, \oplus, !\}$ represents the situation where nothing is known about the relationship between the sets. We call this layer-5. Layer-1 corresponds with the $\mathbb{B}_5$ relations, representing situations where the relationship between the sets is known (*e.g.*, they are equal, one is the subset of the other, etc). Layer-4 nodes represent situations where one relation is known *not* to hold. The bottom of the lattice $\perp$ represents an impossible situation, one which might arise from some operation on sets (*e.g.*, determining the relationship between sets $N$ and $M$ if we know that $N \equiv M$ *and* $N \,!\, M$, an impossible situation if the sets are non-empty).

**The Maximally Informative Relation** (`mir`). For a pair of taxa $N, M$ usually

many of the relations in $\mathbb{R}_{32}$ hold, *i.e.*, evaluate to true. For example, if $N \equiv M$, then for any disjunction $\circ \in \mathbb{R}_{32}$ containing $\equiv$, $N \circ M$ also holds. However, there is a single distinguished true relation which implies *all* other true relations in $\mathbb{R}_{32}$. This is called the *maximally informative relation* (`mir`). Given a `mir` of $\equiv$ between two nodes $N, M$, we know that $N \equiv M$, $N\{\equiv, \subsetneq\}M$, $N\{\equiv, \oplus, !\}M$, etc.

### Other Taxonomic Constraints

So far we have considered fragments of our taxonomy language ($\mathcal{L}_{\mathcal{T}_1}$) for capturing simple `isa`-hierarchies $\leq_{isa}$ (Section 4.3.1), the five basic relations $\mathbb{B}_5$ (Section 4.3.2), and the 32 combined relations $\mathbb{R}_{32}$ (Section 4.3.2).

Many other types of constraints may hold in a taxonomy. Such constraints may be applied to a single taxon in a taxonomy, or to all taxa that apply. Some examples are:

**Non-Emptiness** (N). A given taxon $N \in \mathbf{N}$ of a taxonomy $T = (\mathbf{N}, \leq_{isa}, \mathcal{L}_{\mathcal{T}}, \mathbf{T_C})$ may or may not have actual instances. If we want to express that some nodes $\mathbf{N}_\exists \subseteq \mathbf{N}$ cannot be empty, *i.e.*, have at least one instance, then we can express this using a logic constraint: for each $N \in \mathbf{N}_\exists$, add the constraint:

- $\exists x$: $N(x)$

**Sibling Disjointness** (D). Most biological taxonomies require disjointness of *sibling* elements, *i.e.*, among the children $\{N_1, N_2, \dots\}$ of a parent $M$. For example, if a wolf is a kind of Canis, and a fox is a kind of Canis, then an animal cannot be both a wolf and a fox. The formulas enforcing sibling disjointness are: for each pair of hierarchical constraints $N_1 \leq_{isa} M$ and $N_2 \leq_{isa} M$ with $N_1 \neq N_2$, add the $\mathcal{L}_{\text{MFOL}}$ constraint:

- $\neg\exists x$: $N_1(x) \wedge N_2(x)$

**Coverage** (C). Let $M$ be a taxon whose children are $\{N_1, \dots, N_\ell\}$. We know at least that $N_i \leq_{isa} M$ for all $i = 1, \dots, \ell$. So the union of all children is contained in the parent

$M$. However, we may or may not know whether the converse is true, *i.e.*, that the union of children *covers* the parent. To enforce this, let $\mathbf{M}_\forall \subseteq \mathbf{N}$ be the set of parents who are completely covered by their children (*i.e.*, there is no elements in $M$ that is not also in at least one of its children): for all $M \in \mathbf{M}_\forall$ having children $N_1, \ldots, N_\ell$ of $M$, add:

- $\forall x$: $M(x) \rightarrow \ N_1(x) \vee N_2(x) \vee \cdots \vee N_\ell(x)$

**Further Constraints.** The above constraints are just a few of the many possible constraints that may be assumed for, or required of, a particular taxonomy. Another example constraint might attribute *rank* to taxa.[4] If $T = (\mathbf{N}, \leq_{isa}, \mathcal{L}_\mathcal{T}, \mathbf{T_C})$ is ranked then there is a function $\varrho : \mathbf{N} \rightarrow \mathbf{R}$, where $\mathbf{R}$ is a set of ranks (*e.g.*, $\mathbf{R} = \{\mathsf{species}, \mathsf{genus}, \mathsf{family}, \mathsf{order}, \mathsf{class}, \mathsf{phylum}, \mathsf{kingdom}\}$ is common). These ranks are ordered and every hierarchical constraint $N \leq_{isa} M$ must maintain that order, *i.e.*, $\varrho(N) \leq \varrho(M)$.

### Global Defeasible Constraints Versus Local Constraints

Some of the constraints described above, notably Non-Emptiness $\mathsf{N}$, Sibling Disjointness $\mathsf{D}$, and Coverage $\mathsf{C}$, may be applied to all applicable taxa in a taxonomy, or may be applied selectively. There are reasons to do either. To ensure that a taxonomy being creating adheres to a specific definition of taxonomy, it may make sense to apply the relevant constraints globally. The constraints to apply will depend on the definition of taxonomy.

For example, in a published expert biological taxonomy, each taxon (*e.g.*, *Ranunculus* according to Benson, 1948) has at least one instance (*e.g.*, a species holotype). Each taxon in such a taxonomy has a rank, and the child taxa of a given taxon are disjoint. For any given taxon, the expert usually has tried to capture the full extent of the taxon (perhaps within a geographic scope); the set of a taxon's elements is defined as the combined elements of that taxon's child taxa. Taken together, we would say that a published taxonomy conforms

[4] The term "rank" is heavily overloaded: In graph theory, the rank of a node is the number of children of that node. In set theory, the rank of a set is the number of elements in the set. Here, a rank is a label attached to taxa.

to the following constraint types: *non-emptiness*, *sibling disjointness*, *coverage*, and *ranks*.

As another example, consider a museum's specimen collection. In this situation, there may be specimens that are not identified down to the species level. If this is the case, a specimen might be an element of a genus-level taxon, but not an element of any child species-level taxon [Blu07]. Therefore, a genus may contain specimens not contained in any of the species under that genus. We would say that the taxonomy does not adhere to the *coverage* constraint. The taxonomy is otherwise similar to the one described above: all taxa in the collection have at least one example (*non-emptiness*), sibling taxa are *disjoint*, and the taxa are *ranked*.

Thus, when formalizing a taxonomy, it is important to know what type of taxonomy is intended. A taxonomist who wishes to produce a traditional taxonomic revision of a given genus would provide a *ranked*, *sibling disjoint*, *covered* taxonomy, requiring *non-emptiness* of taxa.

In all of the above cases, it may be the case that a constraint may not apply to some set of taxa. In this use case, the constraint would apply globally, but would be defeasible locally if it introduced a contradiction. The remainder of this chapter assumes that the $\mathsf{N}$, $\mathsf{D}$, and $\mathsf{C}$, when applied, are applied globally and will be called *global taxonomic constraints* (GTCs).

In addition to using the constraints to ensure that an instance of a taxonomy conforms to the rules of a specified type of taxonomy, the constraints provide additional formulas that may contribute extra deductive power when deciding whether or not an alignment between two taxonomies is logically consistent. The additional formulas may also help a logic-based reasoner deduce additional articulations between taxonomies. Before demonstrating this, we must first define how one may express articulations between different taxonomies as constraints.

## 4.4 Formalizing Articulations as Monadic First-Order Logic Constraints

The previous sections considered individual taxonomies, but not relationships between two or more *different* taxonomies. Borrowing terminology from the area of knowledge representation and formal ontologies, we call a logic constraint relating taxa or concepts from different taxonomies an *articulation* [MWK00].

As defined in Chapter 3, an articulation is a 4-tuple $A = (\mathbf{N}, \mathbf{N}, \mathcal{L}_\mathcal{A}, A_C)$ consisting of two sets of taxa (one from each of two taxonomies), a language for expressing inter-taxonomic constraints $\mathcal{L}_\mathcal{A}$, and a set of inter-taxonomic constraints $A_C$ in that language. The articulation language $\mathcal{L}_\mathcal{A}$ may be the same as the taxonomy language ($\mathcal{L}_\mathcal{T}$) of one of the taxonomies (or of both, if they use the same language), or it may be completely different. In order to work within a reasoning framework, however, all of the languages involved in an articulation, the two $\leq_{isa}$ constraint languages, the two $\mathcal{L}_\mathcal{T}$ languages, and the $\mathcal{L}_\mathcal{A}$ language, must be translatable to a common formal language.

For example, let $T_1$ and $T_2$ be taxonomies, and $\alpha$ an articulation between them, denoted $T_1 \sim_\alpha T_2$. As long as all the constraints in all these artifacts can be represented in, for example, monadic first-order logic, we can check, using an automated reasoner, whether a specific formula $\varphi$ is a consequence of the articulation, *i.e.*, whether $\Phi_{T_1} \cup \Phi_{T_2} \cup \Phi_\alpha \models \varphi$ holds. Some consequences $\varphi$ may be unintended, *e.g.*, if $\varphi = \mathsf{False}$ can be deduced,[5] then $\Phi_{T_1} \cup \Phi_{T_2} \cup \Phi_\alpha$ is inconsistent, indicating that the taxonomies and/or the articulation $\alpha$ "have problems."

As with $\mathcal{L}_{\mathcal{T}_1}$, the language used to define $\mathcal{L}_{\mathcal{A}_1}$ could be first-order logic, monadic first-order logic, or some other formal language. In this chapter, $\mathcal{L}_{\mathcal{A}_1}$ is defined as the subset of $\mathcal{L}_{\mathcal{T}_1}$ that models the $\mathbb{R}_{32}$ constraints. This restricts articulations such that they may only apply between single taxa within the given taxonomies. This is in keeping with the domain chosen to demonstrate the framework, as well as with much of the current work on

[5] Instead of $\mathsf{False}$, sometimes the empty clause $\square$ is used.

alignment [Ehr07]. Once a set of articulations has been cast into $\mathcal{L}_{\mathcal{A}_1}$, it can be converted to $\mathcal{L}_{\text{MFOL}}$ using the same translations as described for $\mathcal{L}_{\mathcal{T}_1}$.

## 4.5 Combining $\leq_{isa}$, $\mathcal{L}_{\mathcal{T}}$, and $\mathcal{L}_{\mathcal{A}}$ into $\mathcal{L}_{\text{tax}}$

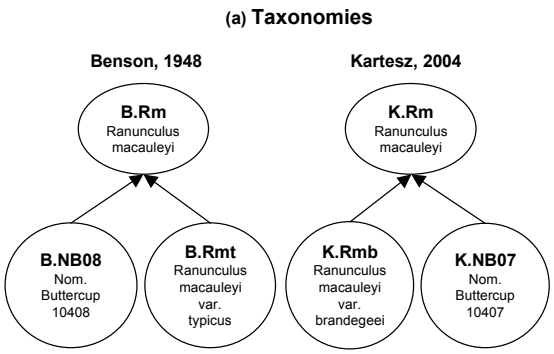
The previous sections defined three different types of relations in an alignment: $\leq_{isa}$, the partial order describing hierarchical relations between taxa in a taxonomy; $\mathcal{L}_{\mathcal{T}}$, the language for describing other constraints within a taxonomy; and $\mathcal{L}_{\mathcal{A}}$, the language for stating articulations between taxa in different taxonomies. The combination of these representations is called $\mathcal{L}_{\text{tax}}$ ($\mathcal{L}_{\text{tax}}$= ($\leq_{isa}$, $\mathcal{L}_{\mathcal{T}}$, $\mathcal{L}_{\mathcal{A}}$)). In this chapter, each element of the triple ($\mathcal{L}_{\text{tax}_1}$ = ($\leq_{isa}$, $\mathcal{L}_{\mathcal{T}_1}$, $\mathcal{L}_{\mathcal{A}_1}$)) has a translation into a common formal language, $\mathcal{L}_{\text{MFOL}}$.

## 4.6 Applying the CleanTax Framework

The remainder of this chapter applies the formalisms described above, and in Chapter 3, to taxonomies and articulations taken from a real-world dataset. The dataset contains seven different expert taxonomies of the genus *Ranunculus* – a flowering plant genus which includes buttercups. In addition to the seven taxonomies, an expert, Peet [Pee05], has asserted many articulations. These articulations are stated using relations drawn from the TCS standard [TCS] for representing taxonomic information. We have translated a subset of those relations into the $\mathbb{R}_{32}$ relations. The following sections describe some applications of the CleanTax framework to this dataset, demonstrating how the framework detects consistent alignments, inconsistent alignments, and new articulations, in small and large taxonomies.

### 4.6.1 Small-Scale Applications of CleanTax

This section uses small taxonomies to demonstrate how the CleanTax framework can detect consistent alignments, inconsistent alignments, and new articulations. The subsequent




(c) $\mathcal{L}_{tax_1}$ Formulas

| Authority | Formula |
|---|---|
| Benson, 1948 | B.NB08 $\leq_{isa}$ B.Rm<br>B.Rmt $\leq_{isa}$ B.Rm |
| Kartesz, 2004 | K.Rmb $\leq_{isa}$ K.Rm<br>K.NB07 $\leq_{isa}$ K.Rm |
| Peet, 2005 | B.Rm $\equiv_P$ K.Rm<br>B.NB08 $\equiv_P$ K.Rmb<br>B.Rmt $\equiv_P$ K.NB07 |
| Non-Emptiness | **N**(B.Rm), **N**(B.NB08)<br>**N**(B.Rmt), **N**(K.Rm)<br>**N**(K.Rmb), **N**(K.NB07) |
| Sibling Disjointness | **D**(B.Rm), **D**(B.NB08)<br>**D**(B.Rmt), **D**(K.Rm)<br>**D**(K.Rmb), **D**(K.NB07) |
| Coverage | **C**(B.Rm), **C**(B.NB08)<br>**C**(B.Rmt), **C**(K.Rm)<br>**C**(K.Rmb), **C**(K.NB07) |

Figure 4.3: Equivalence between (a) two taxonomies, (b) given articulations, (c) represented in $\mathcal{L}_{\text{tax}_1}$ formulas.

section describes a more large-scale application of CleanTax.

### Determining the Consistency of an Alignment

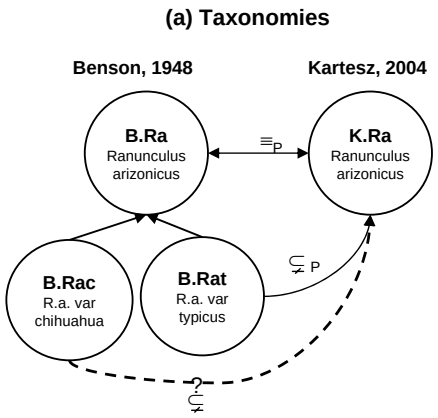
Figure 4.3 shows the equivalence between two taxonomies for the species *Ranunculus macauleyi* and its varieties. Part (a) of the figure shows a taxonomy published by Benson [Ben48], and a newer one, published by Kartesz [Kar04]. Part (b) lists the articulations between the taxa provided by Peet. Part (c) combines the taxonomies and the articulations into statements in $\leq_{isa}$, $\mathcal{L}_{\mathcal{T}_1}$, and $\mathcal{L}_{\mathcal{A}_1}$. These statements, combined with the N, D, and C GTCs are then transformed into $\mathcal{L}_{\text{MFOL}}$ formulas, shown in Table 4.1.

To prove the consistency of the alignment in Figure 4.3, we can apply an automatic reasoner such as Mace4[6] to the $\mathcal{L}_{\text{MFOL}}$ sentences in Table 4.1. Mace4 discovers models that satisfy sets of first-order logic formulas. If Mace4 finds such a model, the formulas must be consistent. In the case of Figure 4.3, Mace4 finds a model with two domain elements, $a$ and $b$: $a$ satisfies $B.NB08(x)$, $B.Rm(x)$, $K.Rmb(x)$, and $K.Rm(x)$, while $b$ satisfies $B.Rmt(x)$, $B.Rm(x)$, $K.NB07(x)$ and $K.Rm(x)$. By discovering a model for the

[6] Prover9 and Mace4: http://www.cs.unm.edu/~mccune/mace4/

| **Authority** | **Formula** |
|---|---|
| Benson, 1948 | $(\forall x\text{: } B.NB08(x) \leftrightarrow B.Rm(x)) \vee$<br>$\quad (\forall x\text{: } B.NB08(x) \rightarrow B.Rm(x) \wedge \exists a\ B.Rm(a) \wedge \neg B.NB08(a))$<br>$(\forall x\text{: } B.Rmt(x) \leftrightarrow B.Rm(x)) \vee$<br>$\quad (\forall x\text{: } B.Rmt(x) \rightarrow B.Rm(x) \wedge \exists a\ B.Rm(a) \wedge \neg B.Rmt(a))$ |
| Kartesz, 2004 | $(\forall x\text{: } K.Rmb \leftrightarrow K.Rm(x)) \vee$<br>$\quad (\forall x\text{: } K.Rmb(x) \rightarrow K.Rm(x) \wedge \exists a\ K.Rm(a) \wedge \neg K.Rmb(a))$<br>$(\forall x\text{: } K.NB07(x) \leftrightarrow K.Rm(x)) \vee$<br>$\quad (\forall x\text{: } K.NB07(x) \rightarrow K.Rm(x) \wedge \exists a\ K.Rm(a) \wedge \neg K.NBO7(a))$ |
| Peet, 2005 | $\forall x\text{: } B.Rm(x) \leftrightarrow K.Rm(x)$<br>$\forall x\text{: } B.NB08(x) \leftrightarrow K.Rmb(x)$<br>$\forall x\text{: } B.Rmt(x) \leftrightarrow K.NB07(x)$ |
| Sibling Disjointness | $\forall x\text{: } B.NB08(x) \rightarrow \neg B.Rmt(x)$<br>$\forall x\text{: } K.Rmb(x) \rightarrow \neg K.NB07(x)$ |
| Coverage | $\forall x\text{: } B.Rm(x) \leftrightarrow B.NB08(x) \vee B.Rmt(x)$<br>$\forall x\text{: } K.Rm(x) \leftrightarrow K.Rmb(x) \vee K.NB07(x)$ |
| Non-Emptiness | $\exists x\text{: } B.Rm(x)$<br>$\exists x\text{: } B.NB08(x)$<br>$\exists x\text{: } B.Rmt(x)$<br>$\exists x\text{: } K.Rm(x)$<br>$\exists x\text{: } K.Rmb(x)$<br>$\exists x\text{: } K.NB07(x)$ |

Table 4.1: $\mathcal{L}_{\text{MFOL}}$ rules for Figure 4.3 plus Non-Emptiness, Sibling Disjointness, and Coverage constraints




**(b) Articulations from Peet, 2005**

R. arizonicus (B. 1948) ≡ R. arizonicus (K. 2004)
R.a. var typicus (B. 1948) ⊊ R.arizonicus (K. 2004)

**(c) $\mathcal{L}_{tax}$ Formulas**

| Authority | Formula |
|---|---|
| Benson, 1948 | B.Rac $\leq_{isa}$ B.Ra<br>B.Rat $\leq_{isa}$ B.Ra |
| Peet, 2005 | B.Ra $\equiv_P$ K.Ra<br>B.Rat $\subsetneq_P$ K.Ra |
| Non-Emptiness | **N**(B.Ra), **N**(B.Rac)<br>**N**(B.Rat),**N**(K.Ra) |
| Sibling Disjointness | **D**(B.Ra), **D**(B.Rac)<br>**D**(B.Rat),**D**(K.Ra) |
| Coverage | **C**(B.Ra), **C**(B.Rac)<br>**C**(B.Rat),**C**(K.Ra) |

Figure 4.4: Inference of new articulations from taxonomies (a), articulations from Peet (b), and formulas in $\mathcal{L}_{\text{tax}_1}$ (c)

formulas, Mace4 proves the consistency of the taxonomies and the associated articulations.

### Discovering Unstated Articulations

Leveraging the machinery of first-order logic, we can use the formalisms to discover unstated, but logically implied consequences $\varphi$ (including new articulations) between taxa.

Peet makes the following claims (Figure 4.4): Benson's *Ranunculus arizonicus var. typicus* (marked $B.Rat$) is included in Kartesz's *Ranunculus arizonicus* ($K.Ra$), and Bensons's *Ranunculus arizonicus* ($B.Ra$) is equivalent to Kartesz's *Ranunculus arizonicus* ($K.Ra$). From these facts, it seems plausible that Benson's *Ranunculus arizonus var. chihuahua* ($B.Rac$) must also be contained in Kartesz's *Ranunculus arizonicus.* CleanTax can prove that this is the case.

Intuitively, it is easy to see that $B.Rac$ must be contained in $K.Ra$. $B.Rac$ is a type of $B.Ra$ and Peet says that $B.Ra$ equals $K.Ra$. Because those terms are equivalent, $K.Ra$ may be substituted wherever $B.Ra$ is seen, and therefore, $B.Rac \subseteq_P K.Ra$. To formally prove the hypothesis, CleanTax uses Prover9 on the first-order formulas created by translating the articulations in Figure 4.4 into $\mathcal{L}_{\text{MFOL}}$ along with the $\mathsf{N}$, $\mathsf{D}$, and $\mathsf{C}$ GTCs.

| Clause# | Formula or Clause | Comment |
|---|---|---|
| 1 | $\forall x : B.Rac(x) \rightarrow B.Ra(x)$ | Assumption |
| 3 | $\forall x : B.Ra(x) \leftrightarrow K.Ra(x)$ | Assumption |
| 4 | $\forall x : B.Rat(x) \rightarrow K.Ra(x)$ | Assumption |
| 8 | $\exists x : B.Rat(x)$ | Assumption |
| 10 | $\forall x : B.Rac(x) \rightarrow \neg B.Rat(x)$ | Assumption |
| 12 | $\forall x : B.Rac(x) \rightarrow K.Ra(x) \wedge$ $(\exists y : (K.Ra(y) \wedge \neg B.Rac(y)))$ | Goal |
| 14 | $\forall x : \neg B.Rac(x) \vee B.Ra(x)$ | Clausify 1 |
| 15 | $\forall x : \neg B.Rac(x) \vee \neg B.Rat(x)$ | Clausify 10 |
| 17 | $B.Rac(c6)$ | Deny 12 |
| 18 | $\forall x : \neg K.Ra(c6) \vee \neg K.Ra(x) \vee \neg B.Rac(x)$ | Deny 12 |
| 19 | $B.Rat(c4)$ | Clausify 8 |
| 21 | $\forall x : \neg B.Rat(x) \vee K.Ra(x)$ | Clausify 4 |
| 25 | $\forall x : \neg K.Ra(c6) \vee \neg K.Ra(x) \vee \neg B.Rat(x)$ | Resolve 18 15 |
| 27 | $\forall x : \neg B.Ra(x) \vee K.Ra(x)$ | Clausify 3 |
| 30 | $B.Ra(c6)$ | Resolve 17 14 |
| 35 | $K.Ra(c4)$ | Resolve 19 21 |
| 36 | $\neg K.Ra(c6) \vee \neg K.Ra(c4)$ | Resolve 25 19 |
| 37 | $\neg K.Ra(c6)$ | Copy 36, unit_del 35 |
| 40 | $K.Ra(c6)$ | Resolve 30 27 |
| 41 | $\square$ | Copy 40, unit_del 37 |

Table 4.2: Prover9's proof of the query in Figure 4.4: Is Benson's *Ranunculus arizonicus var. chihuahua* contained in Kartesz's *Ranunculus arizonicus*?

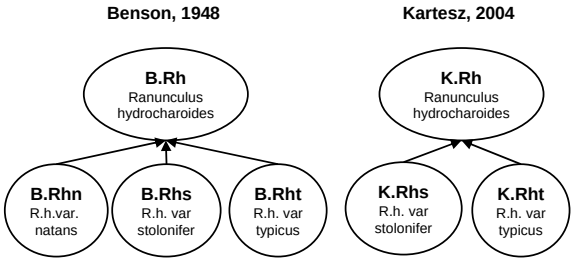
CleanTax then asks Prover9 to prove that $B.Rac$ is a proper subset of $K.Ra$. The proof attempt succeeds with the proof shown in Table 4.2: Prover9 shows that $B.Rac \subsetneq K.Ra$ by proving that the negation of the hypothesis leads to a contradiction. Although the hypothesis seems intuitively true, the automatically derived formal proof involves many mechanical proof steps.

### Detecting Inconsistencies in an Alignment

The CleanTax framework can also detect inconsistencies in an alignment. Consider the alignment from the *Ranunculus* dataset shown in Figure 4.5. The taxonomies and articulation derived from this figure are inconsistent[7] if the taxonomies conform to the N, D, and C

[7] More precisely, $\Phi(T_1) \cup \Phi(T_2) \cup \Phi(A) \models \square$

**(a) Taxonomies**




**(b) Articulations from Peet, 2005**

R. hydrocharoides (B. 1948) ≡ R. hydrocharoides (K. 2004)
R. h. var. stolonifer (B. 1948) ≡ R. h. var. stolonifer (K. 2004)
R. h. var typicus (B. 1948) ≡ R. h. var. typicus (K. 2004)

**(c) $\mathcal{L}_{tax}$ Formulas**

| Authority | Formula |
|---|---|
| Benson, 1948 | $B.Rhn \leq_{isa} B.Rh$<br>$B.Rhs \leq_{isa} B.Rh$<br>$B.Rht \leq_{isa} B.Rh$ |
| Kartesz, 2004 | $K.Rhs \leq_{isa} K.Rh$<br>$K.Rht \leq_{isa} K.Rh$ |
| Peet, 2005 | $B.Rh \equiv_{P} K.Rh$<br>$B.Rhs \equiv_{P} K.Rhs$<br>$B.Rht \equiv_{P} K.Rht$ |
| Non-Emptiness | **N**(B.Rh), **N**(B.Rhn)<br>**N**(B.Rhs), **N**(K.Rht)<br>**N**(K.Rh), **N**(K.Rhs), **N**(K.Rht) |
| Sibling Disjointness | **D**(B.Rh), **D**(B.Rhn)<br>**D**(B.Rhs), **D**(K.Rht)<br>**D**(K.Rh), **D**(K.Rhs), **D**(K.Rht) |
| Coverage | **C**(B.Rh), **C**(B.Rhn)<br>**C**(B.Rhs), **C**(K.Rht)<br>**C**(K.Rh), **C**(K.Rhs), **C**(K.Rht) |

Figure 4.5: An alignment that is inconsistent under the non-emptiness, sibling disjointness, and coverage constraints.

GTCs. We can see the inconsistency of this articulation by asserting a member of *Ranunculus hydrocharoides var. natans* (Benson, 1948). If Benson's taxonomy adheres to the Non-Emptiness constraint, then there must be such an element. If there is such an element, then it is also an instance of *Ranunculus hydrocharoides var. natans* (Benson, 1948). Following Peet's mapping, the instance must also be an instance of *Ranunculus hydrocharoides var. natans* (Kartesz, 2004). According to the Coverage constraint, this means that it must also be an instance of either *Ranunculus hydrocharoides var. stolonifer* (Kartesz, 2004) or *Ranunculus hydrocharoides var. typicus* (Kartesz, 2004). However, if this is the case, then the instance of *Ranunculus hydrocharoides var. natans* (Benson, 1948) must also be an instance of either *Ranunculus hydrocharoides var. stolonifer* (Benson, 1948) or *Ranunculus hydrocharoides var. typicus* (Benson, 1948). Both cases violate the Sibling Disjointness constraint.

This example is interesting because the inconsistency only appears if the taxonomies adhere to the non-emptiness, sibling disjointness, and coverage constraints. Had any of these constraints not been in effect, the alignment would have been considered consistent.

If the sibling disjointness constraint were not in effect, for example, all the elements of $B.Rhn$ could also be elements of $B.Rhs$ and there would be no inconsistency. If the coverage constraint were not in effect, $K.Rh$ could have elements not contained in $K.Rhs$ or $K.Rht$, and these elements could be the same as those in $B.Rhn$. Finally, if the non-emptiness constraint were not in place, then $B.Rhn$ could simply have been empty. The dependence of the proof of inconsistency on the application of all three constraints highlights the importance of clearly stating the type of taxonomies being aligned.

We formally show the inconsistency of the elements in Figure 4.5 using Prover9. To show that $\Phi \models \varphi$ the system adds the negated goal $\neg\varphi$ to the assumptions $\Phi$, trying to find a refutation, *i.e.*, showing that $\Phi \cup \{\neg\varphi\} \vdash \square$ holds.[8] The proof establishing the inconsistency of the articulation in Figure 4.5 appears in the Appendix (A.4.2).

In this example, the explanation for Peet's mapping derives from the geographic scope of the two authorities being mapped. Benson, 1948 provides a world-wide taxonomy for *Ranunculus hydrocharoides* while Kartesz's taxonomy is restricted to North America, where there are no known instances of *Ranunculus hydrocharoides var. natans*. In one sense, limiting the scope to North America, Benson and Kartesz have equivalent definitions of *Ranunculus hydrocharoides*, hence Peet's equivalence mapping. In another sense, when applied globally, Kartesz's notion of *Ranunculus hydrocharoides* should be represented as a subset of Benson's *Ranunculus hydrocharoides*. This indicates that data classified with Kartesz's North American taxonomy may be correctly merged into data classified according to Benson's world taxonomy, but not the other way around. Merging specimens classified with Benson's world taxonomy into a dataset classified with Kartesz's taxonomy, and stating that the combined data are in accordance with Kartesz's taxonomy would lead one to incorrectly believe that Kartesz's taxonomy was global in scope rather than local.

[8] If Prover9 is simply given a set of formulas, a proof means finding a *refutation* of the given set of clauses (any one of which could be thought of as the negated theorem to be proven).

### 4.6.2 A Large-Scale Application of CleanTax

The previous section demonstrated the application of the CleanTax formalizations to small alignments, involving few taxa and articulations. The entire Ranunculus dataset, however, contains far more information. The biggest two taxonomies in the dataset comprise 360 taxa (218 for Benson and 142 for Kartesz), and 206 articulations from Peet. Analyzing taxonomies of this size requires a framework for executing a large number of logical tests. A high-level description of an implementation of the CleanTax framework for this purpose is provided here, followed by results derived from applying the framework to the aforementioned alignment.

#### Target Use Case

This large-scale analysis was carried out in order to address the following use case: Imagine a scientist who wants to integrate data from a number of field studies about observations of Ranunculi around the world. The scientist discovers that some of the studies used a definition of Ranunculus provided by Benson, 1948, and other studies used a more recent definition provided by Kartesz, 2004. In order to harmonize the data, the scientist needs to know how the concepts of Ranunculi compare in these two taxonomies. Luckily, the Peet, 2005 dataset is available for download.

However, it is unclear whether the taxonomies in that dataset meet the scientist's notion of taxonomy. The Peet dataset does not encode taxonomic constraints that the scientist might assume must hold in a biological taxonomy, such as non-emptiness, sibling disjointness, and coverage. So, in order to test the adequacy of the dataset, the scientist should "try out" these constraints and see whether or not the data set adheres to them. If it does, then Peet's articulations may be added and the ensemble may be checked for consistency under these same assumptions.

To fully test the CleanTax system, a complete analysis of the taxonomies, articulations, and added taxonomic constraints is performed and presented to the scientist. This

report presents information describing which combinations of additional assumptions lead to inconsistencies in the taxonomies, or articulations, and, for those assumptions that do not lead to inconsistencies, whether there are any unstated articulations that may be inferred. These unstated articulations may be informative; they may represent new knowledge, or they might highlight problems in the data set.

### The Basic Algorithm

CleanTax begins with a formalization of the given taxonomies and expert articulations as already described. It then converts the taxonomies and articulations into $\mathcal{L}_{\text{MFOL}}$ permitting the application of FOL reasoning techniques to automatically detect inconsistencies and to infer missing articulations via a deductive closure. CleanTax then determines the consistency of the taxonomies and articulations under each combination of additional taxonomic constraints.

Given a pair of taxonomies, a set of articulations between pairs of taxa, and a set of additional taxonomic constraints to test, the **basic algorithm** $\mathcal{A}^0$ is provided in Figure 4.7 (see also Fig. 4.6). The algorithm first finds which of the GTC sets of interest (`gtcSet`) are consistent for each taxonomy. With three GTCs, there are $2^3 = 8$ possible GTC sets (*e.g.* {Non-emptiness}, {Sibling disjointness}, {Non-emptiness, Sibling Disjointness}). The algorithm then finds which of the GTC sets are consistent for the provided articulations (`articulations`). If there are more than two taxonomies, there will be more than one set of articulations (one set between each pair of taxonomies). Finally, the algorithm checks which of the GTC sets are consistent for the combined taxonomies and articulations. The list of GTC sets to test, `gtcs`, is the intersection of the above sets of GTC sets.

Then, each pair of nodes in the goal set is tested for each relation in the set of relations to test, under each GTC set condition. The result of the test can either be `true`, the relation holds, `false`, the relation does not hold, or `unclear`, which can occur if the prover times out or is incomplete. In the basic algorithm, a general theorem prover (*e.g.*, Prover9 [W.W08]) is used for all proofs.

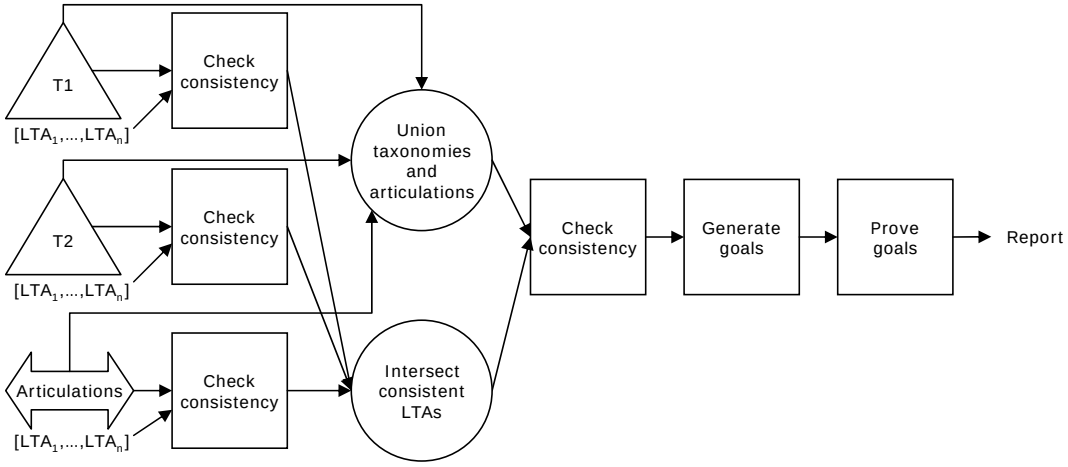



Figure 4.6: Basic CleanTax Methodology

This "brute force" application of the CleanTax framework results in many proof obligations. Consider determining the deductive closure under just one set of assumed constraints, and only deriving the closure for articulations (rather than for articulations *and* unstated intra-taxonomic relations). This would require inspecting at least 218 taxa in Benson $\times$ 142 taxa in Kartesz $\times$ 30 $\mathbb{R}_{32}$ relations (excluding $\top$ and $\bot$) = 928,642 articulations.

### Large-Scale Analysis Results

**Consistency.** The two taxonomies and their articulations were only consistent under the non-emptiness GTC (N) and no-GTC conditions. The coverage GTC (C) alone introduced inconsistencies, so any GTC combination involving C was also inconsistent. The sibling disjointness GTC (D) introduced so many new axioms that neither Mace4 nor Prover9 could process the input with the given resources. Indeed, without GTCs there were 428 logic axioms, while adding the sibling disjointness GTC D yielded a total of 18,104 axioms, most of the form $N(x) \rightarrow \neg\ M(x)$. This demonstrates that while reasoning in $\mathcal{L}_{tax_1}$ may be decidable, it is not necessarily tractable.

**Discovered Articulations.** Table 4.3 shows the counts of `mir` values (see section 4.3.2) of all relationships given and discovered under the two consistent GTC sets. Note first

**Algorithm $\mathcal{A}^0$**

*Input:* `gtcSet` is the set of GTC combinations we want to test, `combinedAxioms` is the set of formulas generated from the alignment, `articulationSet` is the set of given articulations, `goals` is a set of pairs of nodes to test relations between, `relations` is a set of relations to test, `taxonomies` is the set of taxonomies in the alignment, `articulationSet` is the set of articulations - one for each pair of taxonomies.
*Output:* a set of proof results (true, false, unclear), one for each relation tested for each pair of goal pairs, under each GTC condition

```
for taxonomy in taxonomies:
  goodGTCSet[taxonomy] = getConsistentGTCs(taxonomy, gtcSet)

for articulationSet in articulations:
  goodGTCSet[articulation] = getConsistentGTCs(articulationSet, gtcSet)

goodGTCSet[combined] = getConsistentGTCs(combinedAxioms, gtcSet)

gtcs = intersection(goodGTCSet)
for gtc in gtcs:
  for goal in goals:
    for relation in relations:
        proved = proveGoal(gtc, goal, relation)
        if (not proved):
          counter =
              findCounterExample(gtc, goal, relation)
          if (not counter):
            print "unclear"
          else:
            print "false"
        else:
          print "true"
```

Figure 4.7: Algorithm $\mathcal{A}^0$

that CleanTax found a number of new informative (the top node $\top = \{\equiv, \subsetneq, \supsetneq, \oplus, !\}$ is not considered informative) articulations in both consistent GTC conditions, increasing the number from the 218 provided by Peet to 550 articulations in the no GTC condition and 552 in the N GTC condition. While the original alignment contained mostly $\mathbb{B}_5$ relations (and a few not-$\equiv$ relations), CleanTax discovered a significant number of disjunctive relations. Each of these disjunctive relations can be considered an improvement over the original alignment in which these relations were of the $\{\equiv,\subsetneq,\supsetneq,\oplus,!\}$ type. Each such relation could be brought to the attention of an articulator as an opportunity to increase the specificity of the alignment (more on this in Chapter 9). Finally, CleanTax also found a significant number of additional $\mathbb{B}_5$ relations.

| Relation | Given relations | No GTCs | Type N GTC |
|---|---|---|---|
| $\{\equiv\}$ | 112 | 112 | 112 |
| $\{\subsetneq\}$ | 15 | 137 | 137 |
| $\{\supsetneq\}$ | 63 | 90 | 90 |
| $\{\oplus\}$ | 4 | 4 | 4 |
| $\{!\}$ | 12 | 12 | 12 |
| $\{\equiv,\subsetneq\}$ | 0 | 28 | 28 |
| $\{\equiv,\supsetneq\}$ | 0 | 138 | 138 |
| $\{\supsetneq,\oplus\}$ | 0 | 5 | 5 |
| $\{\supsetneq,\oplus,!\}$ | 0 | 10 | 10 |
| $\{\equiv,\subsetneq,\supsetneq,\oplus\}$ | 0 | 3 | 5 |
| $\{\subsetneq,\supsetneq,\oplus,!\}$ | 12 | 11 | 11 |

Table 4.3: New `mir` relations found under the two consistent GTCs.

### 4.6.3 Modularization via Connected Subgraphs

The inconsistency of the alignment under most GTC combinations did not permit a favorable environment for studying the effects of different global taxonomic constraints on the types of articulations discovered. To obtain a good environment for such an investigation, the given taxonomies and articulations were divided into a set of 81 connected subgraphs, each representing a species in one taxonomy and all the related taxonomic concepts in the other taxonomy. These sub-taxonomies were then tested for consistency under each GTC

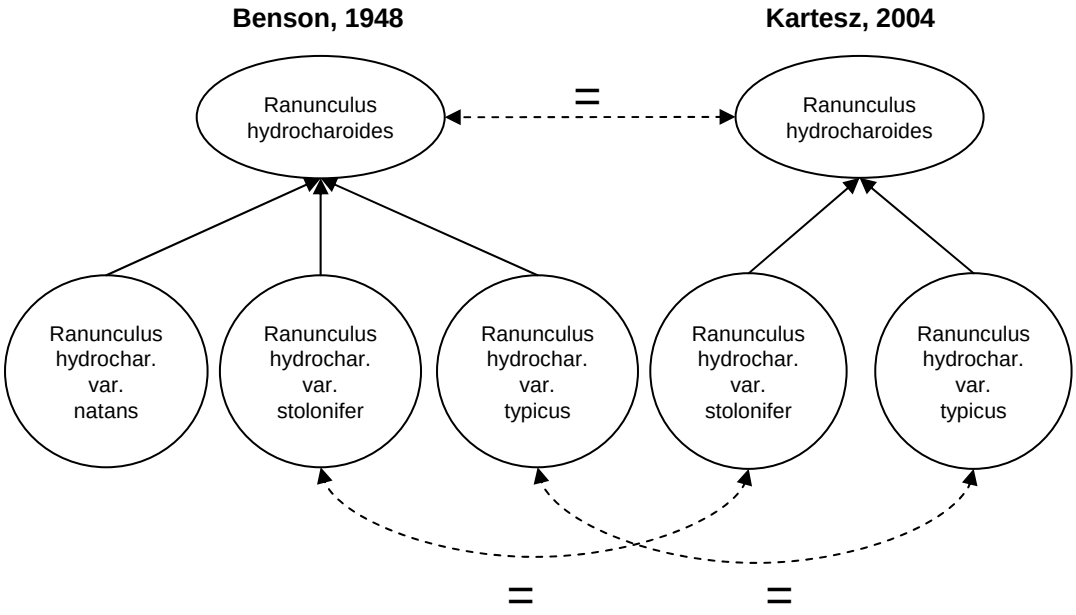



Figure 4.8: Inconsistent set of taxonomies with articulations. Dashed lines are expert articulations.

combination.

Of the 81 sub-taxonomies, 6 were inconsistent under some GTC combination. Figure 4.8 shows an example of a discovered inconsistency under the NDC GTC combination. We can see the inconsistency of this articulation by asserting a member of *Ranunculus hydrocharoides var. natans* (Benson, 1948).

In the 75 sub-taxonomies that were consistent under all GTC combinations, between 357 and 519 new, informative `mir` relations were discovered, depending on the GTC combination. Table 4.4 shows that as more global taxonomic constraints are placed on the taxonomies (*e.g.*, from No-GTCs, to C, to NC, to NCD-GTC combinations), the specificity of discovered relationships also increases. For example, under the No-GTC condition, 357 new, informative `mir` relations were found and 304 of those contained some uncertainty (were disjunctive). In comparison, under the NDC-GTC condition, 519 new, informative, relations were discovered, only 74 of which contained uncertainty.

| | ∅ | N | D | C | ND | NC | DC | NDC |
|---|---|---|---|---|---|---|---|---|
| {≡} | 111 | 111 | 111 | 114 | 111 | 114 | 114 | 114 |
| {⊊} | 33 | 33 | 38 | 33 | 64 | 33 | 38 | 64 |
| {⊋} | 96 | 96 | 105 | 107 | 137 | 107 | 116 | 148 |
| {⊕} | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| {!} | 0 | 0 | 134 | 0 | 134 | 0 | 144 | 144 |
| {≡,⊊} | 31 | 31 | 26 | 31 | 0 | 31 | 26 | 0 |
| {≡,⊋} | 44 | 44 | 35 | 42 | 3 | 42 | 33 | 1 |
| {⊕,!} | 0 | 0 | 0 | 12 | 0 | 12 | 5 | 5 |
| {⊊,⊕,!} | 0 | 0 | 0 | 5 | 2 | 5 | 0 | 0 |
| {⊋,⊕,!} | 17 | 17 | 17 | 38 | 45 | 38 | 34 | 38 |
| {≡,⊊,⊋,⊕} | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
| {⊊,⊋,⊕,!} | 20 | 20 | 20 | 4 | | 4 | 0 | 0 |
| {≡,⊊,⊋,⊕,!} | 192 | 190 | 58 | 158 | 46 | 158 | 34 | 30 |

Table 4.4: New `mir` relationships for each GTC combinaton in the 75 sub-taxonomies that are consistent under the NDC-GTC combination.

## 4.7 Contributions and Future Work

This chapter has provided a framework for a monadic first-order logic representation of taxonomies and articulations between them. The chapter also showed how the GTCs may be used to determine whether or not a given taxonomy conforms to a required type of taxonomy. GTCs may also be used to test the consistency of two taxonomies together with articulations between them. Finally, the chapter showed how GTCs affect the types of relations that may be inferred; as more constraints are added, the inferred relations become more specific.

Unfortunately, the unoptimized version of the CleanTax framework is prohibitively slow. Determining all the articulations between the two large taxonomies described in section 4.6.2 took 46 hours on a fairly unencumbered computer equipped with single-core 3 Ghz Pentium processor. A significant part of the processing time involved overhead from calling the reasoners hundreds of thousands of times. Future work will focus on reducing this overhead. A more theoretically interesting approach to increasing the system's speed is to investigate optimizations that will reduce the number of proof obligations, and to study the impact of trying less expressive, but more tractable logics, such as description logics,

or propositional logic. These are the subjects of the subsequent chapter.

# Chapter 5

# Optimizations

## 5.1 Overview and Objectives

Chapter 4 presented $\mathcal{L}_{\mathcal{T}_1}$, a language for representing taxonomies and articulations using monadic first-order logic, as well as CleanTax, an application of this language in a framework designed to address the questions described in section 1.4. While the language and the framework successfully answered questions about taxonomy and alignment consistency, and were able to infer new articulations, the "tell me everything" use case studied at the end of Chapter 4 demonstrated that the process as described can be prohibitively slow.

This chapter[1] introduces a set of optimizations that increase the efficiency of the framework. These optimizations are of two types: optimizations that reduce the number of proofs necessary to answer a given question, and optimizations that use more tractable representational languages than $\mathcal{L}_{\text{MFOL}}$.

## 5.2 Reducing the Number of Proof Obligations

To reduce the large number of proof obligations in CleanTax, we consider several optimizations involving various lattices.

[1] Much of this chapter is drawn from [Tha08] and [TBL09a].

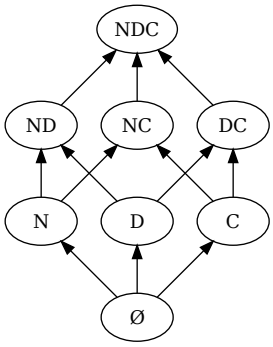



Figure 5.1: A lattice of three global taxonomic constraints (GTCs): non-emptiness N, sibling disjointness D, and coverage, C. When more than one GTC is applied to a taxonomy, the relevant abbreviations are concatenated (*e.g.*, ND when both non-emptiness and sibling disjointness are applied.)

### 5.2.1 GTC Lattice Optimization.

The power set of the three GTCs, N, D, and C, gives rise to a lattice of eight GTC combinations (Figure 5.1) which can be exploited to avoid unnecessary work.

Adding new axioms to a set of formulas already shown to be inconsistent will never result in a consistent set of formulas ($\mathcal{L}_{\text{MFOL}}$ is monotonic). Therefore, once a given GTC is shown to create an inconsistency for a taxonomy or alignment, no parent nodes of that GTC in the GTC lattice need to be investigated.

#### The GTC optimized CleanTax algorithm: $\mathcal{A}$

In the initial CleanTax algorithm (section 4.6.2), if a GTC combination was found to be inconsistent with one taxonomy, it was still tested with the others, and the articulations. The GTC optimized version winnows the set of applicable GTCs down with each test, reducing the number of tests executed and eliminating the need to do an intersection at the end. The optimization affects how `gtcSet` is calculated in $\mathcal{A}^0$ and the changed lines are shown in Figure 5.2.

**Algorithm fragment $\mathcal{A}$:** More efficiently calculating `gtcSet`

```
for taxonomy in taxonomies:
  gtcSet = getConsistentGTCs(taxonomy, gtcSet)

for articulationSet in articulations:
  gtcSet = getConsistentGTCs(articulationSet, gtcSet)

gtcSet = getConsistentGTCs(combinedAxioms, gtcSet)
```

Figure 5.2: Calculating `gtcSet` in $\mathcal{A}$

### 5.2.2 $\mathbb{R}_{32}$ Lattice Optimizations

All of the $\mathbb{R}_{32}$ lattice optimizations take advantage of the fact that the `mir` between two taxa determines the truth of the rest of the relations in the lattice (section 4.3.2).

Some of the optimizations below take advantage of a binary representation of the nodes in the $\mathbb{R}_{32}$ lattice. Each node in the $\mathbb{R}_{32}$ lattice can be assigned a binary value by assigning a 1 to each relation that holds in the disjunction described by the node. As we have five relations, the binary string will be five bits long. The association between place and relation is arbitrary, but for the sake of example, the following order is used: $\equiv, \subsetneq, \supsetneq, \oplus, !$. So, the node in the $\mathbb{R}_{32}$ lattice representing just $\equiv$ would be represented by 10000, and the node representing "not includes" ($\{\equiv,\subsetneq,\oplus,!\}$) would be represented by 11011.

We start with an algorithm for calculating the true $\mathbb{R}_{32}$ relations from a `mir`.

#### Determining the $\mathbb{R}_{32}$ Relations From the Maximally Informative Relation

The algorithm in Figure 5.3 calculates the full set of $\mathbb{R}_{32}$ relations between two taxa given their `mir`. Start with the binary representation of the `mir` and, if the `mir` is not already the top of the $\mathbb{R}_{32}$ lattice, generate the next rank of nodes in the $\mathbb{R}_{32}$ lattice by generating a new set of nodes, each equal to the binary *or* of the `mir` and a binary number which is true at one position (with five relations, that would be the binary representations of 1, 2, 4, 8, and 16). This generates the set of nodes that are true at the next level of the $\mathbb{R}_{32}$ lattice. Then recurse, performing the same action on the newly generated nodes. Continue

until there are no nodes left on the todo list. There will be a great deal of redundancy, so only place new nodes on the todo and done lists if they are not already on the list.

#### Determining the Maximally Informative Relation Top Down: $\mathcal{A}^{\downarrow}_{\texttt{min}}$

This algorithm uses the layer-4 nodes from the $\mathbb{R}_{32}$ lattice to determine the `mir` between two taxa. Again, once `mir` is known for two taxa, the values of the rest of the nodes in the $\mathbb{R}_{32}$ lattice follow. Let $\mathbb{T}_4$ be the layer-4 relations that are true for some pair $N, M$. One can show that $\texttt{mir} = \bigcap_{R \in \mathbb{T}_4} R$. Thus by testing exactly 5 out of the 30 non-trivial $\mathbb{R}_{32}$ relations and combining those results, $\mathcal{A}^{\downarrow}_{\texttt{min}}$ avoids all other 25 tests.

The algorithm for determining the `mir` from the top five relations (Figure 5.4) simply performs the *and* of the binary representations of the top five relations that hold between a given pair of taxa. The result is the binary representation of the `mir` node.

This algorithm for determining the `mir` from the truth values of the five layer-4 nodes in the $\mathbb{R}_{32}$ lattice informs a new version of the core CleanTax algorithm.

**Algorithm** $\mathcal{A}^{\downarrow}_{\texttt{min}}$ (Figure 5.5) is a modification of the slightly modified CleanTax algorithm in section 4.6.2 . Rather than iterating through every relation in $\mathbb{R}_{32}$, $\mathcal{A}^{\downarrow}_{\texttt{min}}$ only tests the five nodes in layer-4 of the $\mathbb{R}_{32}$ lattice.

#### Determining the Maximally Informative Relation Bottom-Up: $\mathcal{A}^{\uparrow}_{\texttt{min}}$

**Algorithm** $\mathcal{A}^{\uparrow}_{\texttt{min}}$ proceeds bottom up, starting at layer-1 in $\mathbb{R}_{32}$ and stopping as soon as a true node is found. In the best case, one of the layer-1 nodes evaluates to true (there are at most 5 proofs for layer-1); in the worst case no layer-4 node evaluates to true, *i.e.*, we know nothing about the relationship between $N$ and $M$. The latter will result in 30 tests in the lattice: recall that we skip tests with $\bot$ (always false) and $\top$ (always true).

### 5.2.3 $\mathbb{R}_{32}$ Lattice Optimization Results

Table 5.1 shows the impact of the two $\mathbb{R}_{32}$ lattice optimizations. As shown in Chapter 4, calculating the deductive closure of two taxonomies and a set of articulations, under even a

**Algorithm Expand `mir`**

*Input:* A `mir` relation, mir
*Output:* A set of relations implied by that `mir` relation

```
doneList = (mir)
todoList = ()
numRanks = 5
trueNodes = getImplicatedRelations(doneList, todoList)

def getImplicatedRelations(doneList, todoList):
    while (relation = pop(todoList)):
        pushIfNew(relation, doneList, doneList)

        if (getRank(relation) < numRanks):
          newTodo = pushIfNew(relation | b10000, newTodo, doneList)
          newTodo = pushIfNew(relation | b01000, newTodo, doneList)
          newTodo = pushIfNew(relation | b00100, newTodo, doneList)
          newTodo = pushIfNew(relation | b00010, newTodo, doneList)
          newTodo = pushIfNew(relation | b00001, newTodo, doneList)

  if (newTodo.length > 0):
    getImplicatedRelations(newTodo, doneList)
  else:
    return doneList

def pushIfNew(item, list1, list2):
  if (not (item in list1) and not(item in list2)):
      push(item, list1)

  return list1
```

Figure 5.3: Finding all relations implied by a `mir` relation

**Algorithm Top-Down `mir` Discovery**

*Input:* `trueLayer4Rels`, the set of layer-4 relations that hold between a pair of nodes
*Output:* The implied `mir` relation

```
xmir = b11111
for relation in trueLayer4Rels:
  xmir = xmir & relation
return xmir
```

Figure 5.4: Finding the `mir` between two nodes based on the truth value of the five layer-4 relations.

**Algorithm $\mathcal{A}^{\downarrow}_{\texttt{min}}$**

*Input:* the only difference between the input of $\mathcal{A}^{\downarrow}_{\texttt{min}}$ and $\mathcal{A}^{0}$ is `topFiveRelations`, the five layer-4 relations.
*Output:* a set of proof results (true, false, unclear), one for each relation tested for each pair of goal pairs, under each GTC condition.

```
for taxonomy in taxonomies:
  gtcSet = getConsistentGTCs(taxonomy, gtcSet)

for articulationSet in articulations:
  gtcSet = getConsistentGTCs(articulationSet, gtcSet)

gtcSet = getConsistentGTCs(combinedAxioms, gtcSet)

for gtc in gtcSet:
  for goal in goals;
    for relation in topFiveRelations: # here is the optimization
        proved = proveGoal(gtc, goal, relation, goalType)
        if (not proved):
          counter =
              findCounterExample(gtc, goal, relation, goalType)
          if (not counter):
            print "unclear"
          else:
            print "false"
        else:
          print "true"
```

Figure 5.5: Algorithm $\mathcal{A}^{\downarrow}_{\texttt{min}}$

**Algorithm $\mathcal{A}^{\uparrow}_{\texttt{min}}$**

*Input:* the same as $\mathcal{A}^0$.
*Output:* a set of proof results (true, false, unclear), one for each relation tested for each pair of goal pairs, under each GTC condition.

```
for taxonomy in taxonomies:
  gtcSet = getConsistentGTCs(taxonomy, gtcSet)

for articulationSet in articulations:
  gtcSet = getConsistentGTCs(articulationSet, gtcSet)

gtcSet = getConsistentGTCs(combinedAxioms, gtcSet)

for gtc in getSet:
  for goal in goals:
  index = 0
  foundMIR = false
  while (index < relations.length) and (not foundMIR)):
      relation = relations[index]
      proved = proveGoal(gtc, goal, relation, goalType)
      if (not proved):
              counter =
                  findCounterExample(gtc, goal, relation, goalType)
        if (not counter):
          print "unclear"
        else:
          print "false"
      else:
        print "true"
        foundMIR = true
     index++
```

Figure 5.6: Algorithm $\mathcal{A}^{\uparrow}_{\texttt{min}}$

| | $\mathcal{A}^0$ | $\mathcal{A}^{\uparrow}_{\texttt{min}}$ | $\mathcal{A}^{\downarrow}_{\texttt{min}}$ |
|---|---|---|---|
| Judgements | 928,680 | 912,779 | 154,780 |
| User time (mins) | 2,768.86 | 2,720.19 | 469.99 |
| System time (mins) | 41.79 | 41.07 | 7.60 |
| Logical steps (millions) | 2,634 | 2,589 | 442 |

Table 5.1: Impact of optimizations on deductive closure under the non-emptiness (N) GTC.

| | $\mathcal{A}^0$ | $\mathcal{A}^{\uparrow}_{\texttt{min}}$ | $\mathcal{A}^{\downarrow}_{\texttt{min}}$ |
|---|---|---|---|
| Judgements | 17,019 | 2,194 | 2,745 |
| User time (secs) | 573.59 | 83.61 | 100.47 |
| System time (secs) | 1,189.59 | 195.91 | 192.36 |
| Logical steps (thousands) | 2,484 | 384 | 394 |

Table 5.2: Impact of optimizations on the deductive closure under the NDC LTA for 75 sub-taxonomies.

single GTC, can involve a great number of logical tests. There is only a slight improvement of the bottom-up algorithm $\mathcal{A}^{\uparrow}_{\texttt{min}}$ over the base algorithm $\mathcal{A}^0$. However, $\mathcal{A}^{\downarrow}_{\texttt{min}}$ reduces the number of tests by 84% and is processed almost 6 times as quickly as the unoptimized $\mathcal{A}^0$.

The populated relation lattice shown in Figure 5.7 shows, for each relation under the N GTC, the number of times the relation was true according to the optimization, the number of times it was found true using a reasoner, and the number of times the relation was the `mir`. This type of visualization using the $\mathbb{R}_{32}$ lattice can be quite useful.

The improvement of $\mathcal{A}^{\uparrow}_{\texttt{min}}$ is small because the N GTC engenders very little deductive power, so in general the relation between any two given nodes is frequently unknown, resulting in the worst-case scenario for $\mathcal{A}^{\uparrow}_{\texttt{min}}$: all 30 relations must be checked. To investigate the impact of the optimizations in a scenario with less uncertainty, the optimizations were run using the 75 consistent sub-taxonomies described in section 4.6.3. As mentioned in that section, the specificity of discovered articulations increases as more GTCs are applied to the taxonomies. Based on this, we would expect an improvement in the $\mathcal{A}^{\uparrow}_{\texttt{min}}$ optimization.

Table 5.2 demonstrates that the $\mathcal{A}^{\uparrow}_{\texttt{min}}$ optimization improves relative to the $\mathcal{A}^{\downarrow}_{\texttt{min}}$ optimization under the NDC GTC combination, which engenders the inference of many specific relations.

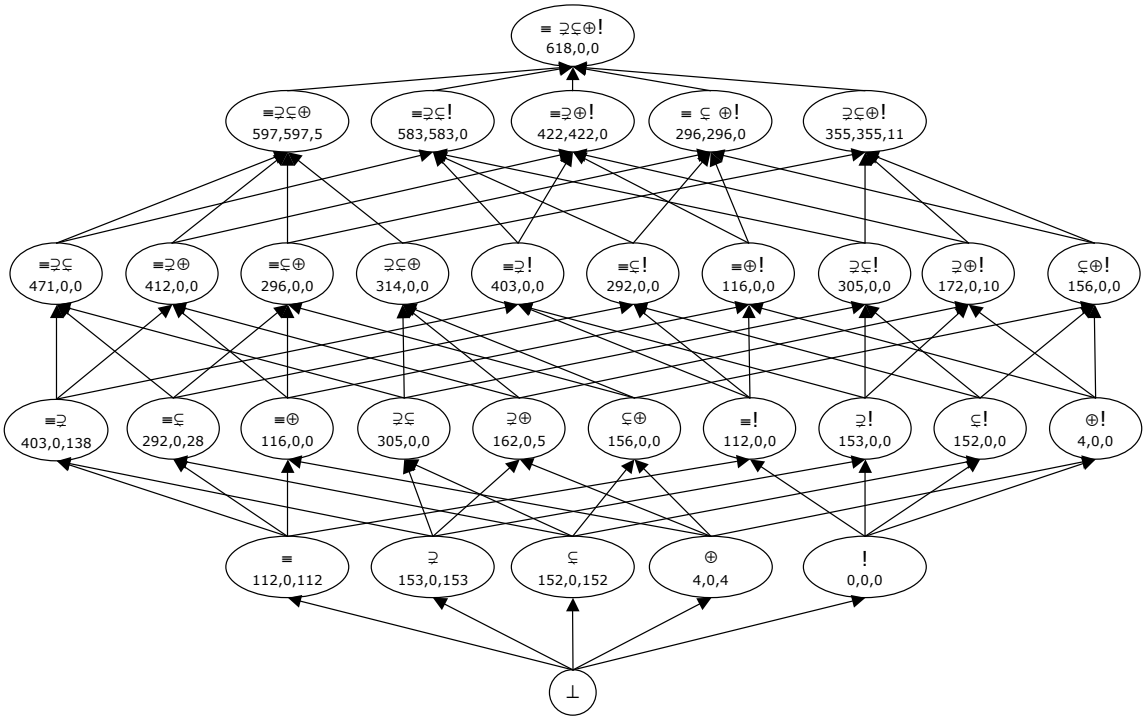



Figure 5.7: The populated relation lattice shows, for each relation under the N GTC, the number of times the relation was true according to the $\mathcal{A}^{\downarrow}_{\texttt{min}}$ optimization, the number of times it was found true using a reasoner, and the number of times the relation was the `mir`.

### 5.2.4 Summary of Lattice Optimizations

The optimizations described in this section go a long way towards boosting the efficiency of CleanTax, dropping the time to calculate the deductive closure of a large alignment from 46 hours to under 8. Even so, 8 hours still seems excessively long. A large part of the time is still spent on overhead involved in calling a reasoner many thousands of times. Reducing this overhead will help, and remains to be done. Another improvement may come from shifting from monadic first-order logic to some less expressive, but more tractable logic.

## 5.3 Language Optimizations

This section begins with a description of language expressiveness. Then it describes experiments with logics less expressive than $\mathcal{L}_{\text{MFOL}}$. Using a less expressive language will restrict the types of taxonomies and articulations that can be represented, and the types of questions that may be asked. However, depending on the applications, these restrictions may not be burdensome, and the calculus used to reason with a less expressive language may afford more rapid reasoning.

### 5.3.1 Expressive Power

A formal language is a set of symbols, a set of formulation rules for combining those symbols into well formed formulas, and often a semantics assigning a meaning to the symbols and well formed formulas [EFT94]. The expressive power of a language is described by the set of sentences it can define. We say that language $\mathcal{L}_\alpha$ is at least as expressive as language $\mathcal{L}_\beta$, denoted $\mathcal{L}_\alpha \geq \mathcal{L}_\beta$ if for every expression in $\mathcal{L}_\beta$ there is an equivalent expression in $\mathcal{L}_\alpha$. Two languages have the same expressive power if $\mathcal{L}_\alpha \geq \mathcal{L}_\beta$ and $\mathcal{L}_\beta \geq \mathcal{L}_\alpha$

The goal of working with less expressive languages is to reduce the complexity of processing within those languages. For example, while determining satisfiability in first-order logic is undecidable, it is decidable in propositional logic, and most Description Logics. Within Description Logics, very restricted languages such as the Frame Language version

$FL^-$ [LB87], and $\mathcal{AL}$ [SSS91] can answer subsumption queries in polynomial time, while small changes to these languages, such as adding role restrictions to $FL^-$ or intersection to $\mathcal{AL}$ [DLNN97], render the problem of answering subsumption queries NP-complete.

### 5.3.2 Complexity

Different languages have different levels of complexity. The complexity of a language is reflected in the amount of time it takes to solve *decision problems* as the size of the input to those problems grows. A decision problem is one that has a yes-no answer, depending on the inputs. For example, a basic decision problem in monadic first-order logic is, given some $\mathcal{L}_{\text{MFOL}}$ formulas $\Phi$ and $\varphi$ does $\Phi \vdash \varphi$. There are many classes of complexity, but this dissertation deals primarily with four: **P**, **NP-complete**, **EXP-complete**, and **NEXP-complete**. The complexity of each of these classes may be defined in terms of either the complexity class **NTIME(f(n))**, which is the set of decision problems that can be solved by a non-deterministic Turing machine using O(f(n)) time and unlimited space, or **DTIME** which is defined similarly, but for deterministic Turing machines. The following definitions of the above complexity classes are listed in order of increasing complexity. Given a problem with input size $n$:

$$P = \bigcup_{k\in\mathbb{N}} \text{DTIME}(n^k)$$
$$NP = \bigcup_{k\in\mathbb{N}} \text{NTIME}(n^k)$$
$$EXP = \bigcup_{k\in\mathbb{N}} \text{DTIME}(2^{n^k})$$
$$NEXP = \bigcup_{k\in\mathbb{N}} \text{NTIME}(2^{n^k})$$

The term *complete* when attached to a given class means that there is a polynomial-time reduction of every other problem in that class to that problem. Monadic first-order logic, for example, has been shown to be NEXP-complete [BGW93]. The languages discussed below, Description Logics, propositional logic, and subsets of the RCC calculus, will have

a lower level of complexity.

### 5.3.3 Description Logics

Description Logic (DL) is a family of concept-based knowledge representation languages. The syntax of all DLs has a set of unary predicates called *concept names*, a set of binary relations called *role names*, and a set of *constructors* used to define concept names and role names. Each variety of DL implements a different set of constructors. The most basic DL is called $\mathcal{AL}$ (Attributive Language) [SSS91]. To give a feel for what a simple DL looks like, the syntax and semantics of $\mathcal{AL}$ are provided in section A.1.3. The key features of $\mathcal{AL}$ are that concepts are defined either by simply stating that something is an atomic concept, or by defining a concept by negating an atomic concept or by intersecting two concepts. Concepts can also be defined as those things fulfilling a certain role. Roles can be defined in two ways: using a value restriction (*e.g.*, "all red things") or through limited existential quantification (*e.g.*, "all things with at least one child"). For example, the concept "parent" could be defined as follows: $Parent = Person \sqcap \exists hasChild.\top$. Note that in $\mathcal{AL}$, the object of the *hasChild* role cannot be restricted, so according to this definition, the child of a *Person* does not have to be a *Person*. In order to assert such a restriction, a more expressive DL must be used (*e.g.*, one called $\mathcal{ALE}$).

DLs can be very expressive, and, unlike full $\mathcal{L}_{\text{FOL}}$, still decidable. Two of the main representations for the semantic web, OWL-DL and OWL-Lite are both varieties of DL, each with a different complexity. The undecidibility of first-order logic has driven many researchers studying the alignment problem to use more tractable, but still fairly expressive Description Logics. For example, MoA [KJH+05] infers equivalence and subsumption articulations in OWL-DL documents using lexical information about concept names. Similarly, OLA [EV04] restricts itself to the OWL-Lite Description Logic.

Despite the popularity of DL, especially in the form of OWL-DL, its utility in the context of this thesis is uncertain for two reasons. First, it is unclear how to model even the basic RCC relations in a tractable way using DL. An attempt by [KG05] models regions as

classes. Each region must be *regular* meaning that it is non-empty, and it must contain all its interior points. For a region $X$, the regularity condition is captured by the axioms $X(x)$ and $X \equiv \exists R.(\forall R.X)$, where R is a symmetric and transitive relation. This encoding was recently implemented in [SS09] using an OWL 2 reasoner (OWL 1 did not support some features needed by the encoding). The authors found that the second axiom in regularity condition above made the translation intractable with even a small number of concepts. In reaction, they proposed a hybrid reasoner that applied traditional RCC algebra reasoning for RCC problems, and DL reasoning in other cases. This approach was also proposed in [GBM07], who provide an excellent review of the various unsatisfactory attempts to encode RCC relationships and reasoning in DL. The second problem for DL is the disjunctive relations used in CleanTax. Neither [KG05] nor [SS09] attempted to encode disjunctive RCC relationships in DL. Although there are fairly clear encodings for the basic relations, it is unclear how to represent disjunctive relations, such as the articulation that two concepts $N, M$ are either equal or they are disjoint. The sentence $(N \equiv M) \sqcup (N \sqcap M \equiv \emptyset)$ is not a syntactically legal sentence in DL. The sentence $MyDisjunct \equiv ((\neg N \sqcup M) \sqcap (N \sqcup \neg M)) \sqcup (\neg M \sqcup \neg N))$ is legal, but means "the concept $MyDisjunct$ represents the collection of instances that are either in both $N$ and $M$ or not in $N$ or not in $M$" which is different from the disjunctive sentence "$N$ and $M$ are either equal or they are disjoint." Stating such a disjunctive sentence in DL is possible, but requires a trick called a *spypoint* [Hor07]. A spypoint is an instance $x$ that is related to every other instance via a given role $p$. The following recipe uses the spypoint to make disjunctive assertions. As an example, consider the situation in which we want to test the disjunctive sentence "$N$ and $M$ are either equivalent or disjoint" given two disjoint concepts $N, M$ $(N \equiv \neg M)$. To state (and test) this disjunction using a spypoint, create two new concepts called Spy and NonSpy. State that $N$ and $M$ are subsumed by NonSpy ($N \sqsubseteq NonSpy$, $M \sqsubseteq NonSpy$). Create a role p, with a domain of Spy and a range of NonSpy. Create another role invP, which is the inverse of p (if p(x,y) then invP(y,x)). Now, assign individuals n, m, and s, to the concepts N, M, and Spy respectively, stating further that n, m, and s are all different

individuals. Also, relate m and n to s via the p role: p(s,n) and p(s,m). With all that in place, we can test disjunctive sentences by defining the Spy concept. For example, defining the Spy concept as $Spy \equiv \forall invP.((\neg N \sqcup M) \sqcap (N \sqcup \neg M))$ will result in an inconsistency, because Spy is being defined as all things satisfying an unsatisfiable property. However, defining the Spy concept as $Spy \equiv \forall invP.(((\neg N \sqcup M) \sqcap (N \sqcup \neg M)) \sqcup (\neg N \sqcup \neg M))$ will not result in an inconsistency, because this property can be satisfied. Needless to say, applying the spypoint trick to even this simple situation is fairly complex. Adding the machinery necessary to quantify uncertainty as described here may reduce any efficiencies afforded by the more tractable language. Furthermore, it is not clear that DL reasoners are any faster than $\mathcal{L}_{\text{FOL}}$ reasoners given the monadic-logic representation in Chapter 4, reducing the benefits of moving to a Description Logic. However, applying the CleanTax framework to another formal language, such as a Description Logic, will demonstrate its usefulness as a benchmarking tool. Researching the possibility of using a DL to represent $\mathcal{L}_{\text{tax}}$ remains an important aspect of future work.

### 5.3.4 Propositional Logic

An even less expressive logic than any Description Logic is propositional logic. [GSY04] described the S-Match ontology alignment framework, which uses propositional logic to infer matches in the schema matching problem. Their articulations were restricted to a subset of the $\mathbb{R}_{32}$ relations, namely $\{\equiv\}, \{\equiv, \subsetneq\}, \{\equiv, \supsetneq\}, \{!\}$ and $\{\equiv, \subsetneq, \supsetneq, \oplus, !\}$. Even with this restricted set of relations, their system was more expressive than the other schema matching systems they describe.

Propositional logic is an attractive possibility from a complexity standpoint. While basic reasoning tasks in monadic first-order logic and OWL-DL are both NEXP-complete, reasoning tasks in propositional logic have an NP-complete complexity and should therefore scale better.

Two propositional encodings for RCC-5 exist: Bennett [Ben94] describes a propositional representation of the $\mathbb{B}_5$ and Pham [PTS06] presents a different propositional encoding for

Allen's interval algebra for temporal relations [All83] that can be modified to apply to RCC-5.

Bennett's representation, described in Table 5.3 covers the $\mathbb{B}_5$ using two sets of formulas - one set describing the formulas that must hold for a given relation, and a second set describing the formulas that must not be entailed by the first set. Bennett's representation does not cover the entire $\mathbb{R}_{32}$.

| Relation | Model Constraint | Entailment Constraints |
|---|---|---|
| X ! Y | $\neg(X \wedge Y)$ | $\neg X, \neg Y$ |
| X $\oplus$ Y | - | $\neg(X \wedge Y)$, $Y \rightarrow X$, $X \rightarrow Y$, $\neg X, \neg Y$ |
| X $\subsetneq$ Y | $(X \rightarrow Y)$ | $Y \rightarrow X$, $\neg X, \neg Y$ |
| X $\supsetneq$ Y | $(Y \leftarrow X)$ | $X \rightarrow Y$, $\neg X, \neg Y$ |
| X $\equiv$ Y | $(X \leftrightarrow Y)$ | $\neg X, \neg Y$ |

Table 5.3: Bennett's [Ben94] propositional representation of $\mathbb{B}_5$

Pham's encoding provides a more complete translation of arbitrary constraint satisfaction problems (CSP). Roughly speaking, given a set of nodes $N$ and a set of relations $R$, the encoding creates a new variable for each $n \times n \times r$ condition ($n \in N, r \in R$). The encoding then adds propositional sentences that constrain the relationships between these nodes in a way that mirrors the original constraint satisfaction problem. Given a CSP of $n$ nodes and $b$ relations, the encoding results in a propositional statement with $O(n^2 * b)$ variables and $O(n^3 * b^2)$ clauses. This increased problem size may lead to poor scaling. Adding axioms for the coverage GTC in this setting is not straightforward. Given a node $A$ and its chldren $B$ and $C$, $A$ is covered by its children if for all known regions $X, PP(X, A) \rightarrow \neg(DC(X, B) \wedge DC(X, C))$ where $PP(X, A)$ is a new variable representing the $\subsetneq$ relation between $X$ and $A$.

Recent work by [WW09] has shown that, at least in the case of networks restricted to $\mathbb{B}_5$ relations, a dedicated RCC reasoner such as GQR [WWG09] will outperform a propositional encoding of an RCC-based problem. However, using these propositional encodings to permit the statement of the coverage constraint may lead to reasoning that can outperform general $\mathcal{L}_{\text{FOL}}$ reasoners. Research addressing this question remains as future work.

### 5.3.5 $\mathbb{R}_5^{28}$: A Tractable Subset of RCC-5

The monadic logic representation described in Chapter 4, as well as the Description Logics representation and propositional logic representation of $\mathbb{R}_{32}$ described above are all decidable, but still have a greater than polynomial-time complexity when answering entailment questions. Renz and Nebel [RN99] showed that there is a subset of $\mathbb{R}_{32}$, called $\mathbb{R}_5^{28}$which can be reasoned over in polynomial time. The subset they describe is the maximal subset of $\mathbb{R}_{32}$ which also contains $\mathbb{B}_5$.

The following theorem by Jonsson and Drakengren [JD97] gives a precise account of the complexity of deciding these questions by identifying all maximal tractable subalgebras of $\mathbb{R}_{32}$.

**Theorem 5.1** ([JD97])**.** Let $\mathbb{R} \subseteq \mathbb{B}_5$ be any of the 32 subsets of $\mathbb{B}_5$. Then deciding whether $\Psi^{\mathbb{R}}$ is satisfiable is **polynomial** iff $\mathbb{R}$ is a subset of one of $\mathbb{R}_5^{28}, \mathbb{R}_5^{20}, \mathbb{R}_5^{17}$, or $\mathbb{R}_5^{14}$, and **NP-complete** otherwise.

Thus, in general we can expect reasoning with combined relations from $\mathbb{R}_{32}$ to be efficient for large sets of constraints only if we consider combined relationships that correspond to one of the four maximal classes $\mathbb{R}_5^{28}$, $\mathbb{R}_5^{20}$, $\mathbb{R}_5^{17}$, or $\mathbb{R}_5^{14}$ (or subsets of those); for other constraint sets over $\mathbb{R}_{32}$, not falling under those four, answering entailment questions becomes an NP-complete problem.

For our purposes, the subalgebra $\mathbb{R}_5^{28}$ (see Appendix A.2) is the most interesting: it contains 28 out of the 32 possible relations in $\mathbb{R}_{32}$. The only *excluded* relations are $\{\subsetneq, \supsetneq\}$, $\{\subsetneq, \supsetneq, \equiv\}$, $\{\subsetneq, \supsetneq, !\}$, and $\{\subsetneq, \supsetneq, \equiv, !\}$. Note that all four of these contain $\{\subsetneq, \supsetneq\}$, *i.e.*, the disjunction $(N \subsetneq M) \vee (N \supsetneq M)$. In practical settings it does not appear to be common that one is uncertain whether $(N \subsetneq M)$ or rather $(N \supsetneq M)$. For example, if the taxonomic ranks of $N$ and $M$ are known, then one of these cases can always be eliminated. The excluded cases could potentially arise as (intermediate) results of a larger reasoning problem. [2]

Summarizing, we obtain:

[2] Note, however, that the excluded cases did not arise in Tables 4.3 and 4.4.

**Corollary 5.2.** Reasoning with arbitrary $\mathcal{L}_{\mathcal{T}_1}$ taxonomy constraints can be infeasible (NEXPTIME-complete) in general. Reasoning with $\mathbb{R}_5^{28}$ taxonomy constraints is efficient (*i.e.*, deciding satisfiability is in polynomial time).

Given these observations, it should be possible to optimize the reasoning problems in CleanTax even more than was possible with the shift to propositional logic.

### 5.3.6 Optimization Results

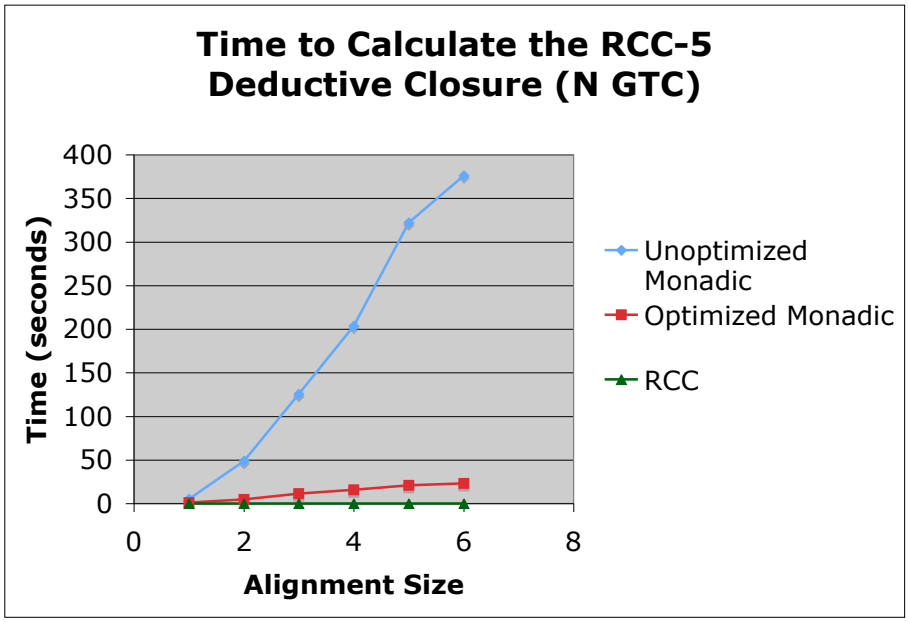



Figure 5.8: Time in seconds to determine RCC-deductive closure for the 75 sub-taxonomies under the N GTC for the unoptimized monadic logic, optimized monadic logic, and RCC-algebra reasoners.

Figure 5.8 compares the amount of time necessary to determine the RCC deductive closure between sub-taxonomies of various sizes under the non-emptiness GTC under different reasoning scenarios. For each sub-taxonomy, the RCC deductive closure in this case was restricted to calculating the `mir` for each possible articulation. The alignment size is the number of such articulations divided by 10. As can be seen in the figure, the unoptimized monadic logic version takes significantly longer than the other two conditions.

The $\mathcal{A}^{\downarrow}_{\texttt{min}}$ optimized version fares considerably better. Both monadic logic versions used the iProver [Kor08] first-order theorem prover. Comparisons between several current first-order theorem provers showed iProver to perform the best for the types of logic problems involved here. GQR [WWG09], currently the fastest RCC reasoner was chosen for the RCC condition.

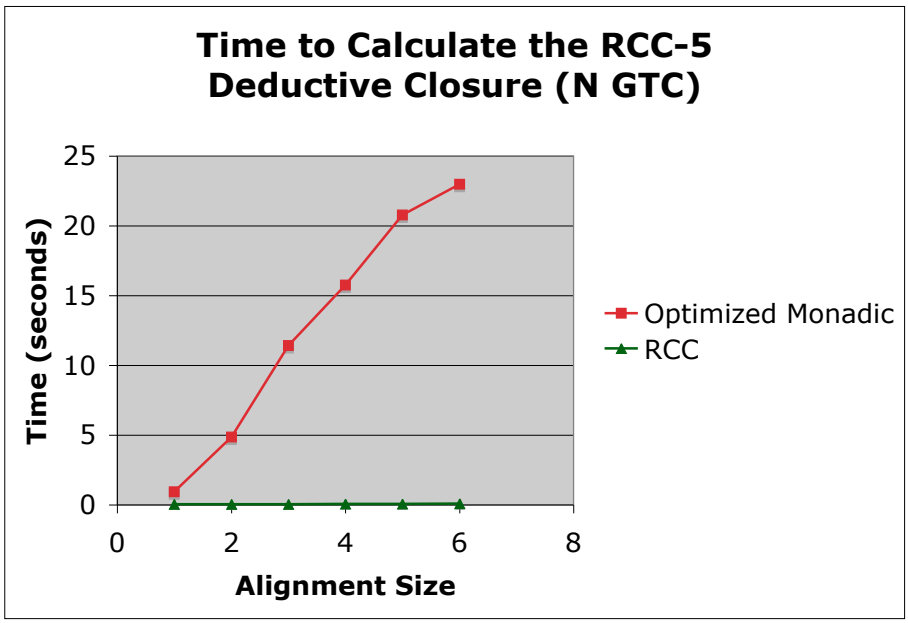



Figure 5.9: Time in seconds to determine RCC-deductive closure for the 75 sub-taxonomies under the N GTC for the optimized monadic logic, and RCC-algebra reasoners.

Figure 5.9 removes the unoptimized monadic logic results so that the optimized monadic logic results can be better compared to the RCC results. Figure 5.10 shows how well the RCC reasoner scales with larger networks. Clearly, when an RCC reasoner can be applied (only in the N and ND GTC conditions), it is far superior to the full first-order reasoner.

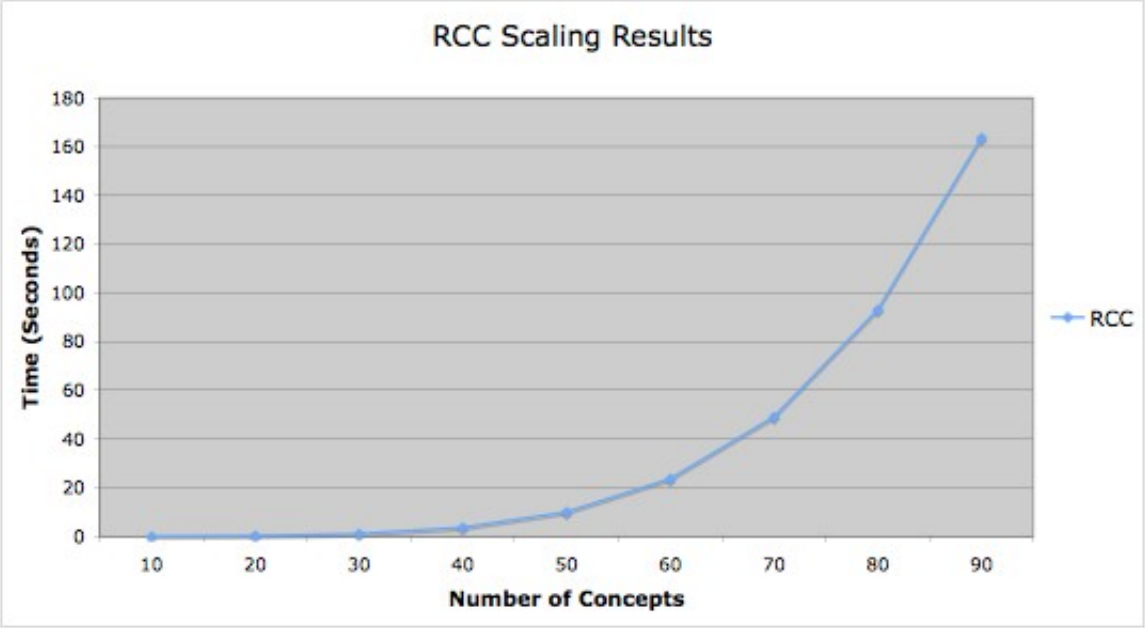



Figure 5.10: Time in seconds to determine RCC-deductive closure for the RCC-algebra using large alignments under the N GTC.

## 5.4 Parallelization

As has been demonstrated, calculating the deductive closure of an alignment can involve many small, independent, reasoning tasks. The independence of these tasks presents an opportunity to process them in parallel. Another kind of optimization would be to develop algorithms for determining the best way to allocate reasoning tasks across a computer cluster. The current implementation of the CleanTax system includes mechanisms for utilizing a cluster using the Sun Grid Engine [Gen01]. However, the current algorithms for dividing tasks among the cluster nodes do not leverage the parallelism very well. Future work will investigate optimal strategies for deploying multiple independent reasoning tasks across a cluster of nodes.

| Language | Complexity | Citation |
|---|---|---|
| First-order logic | undecidable | [Chu36, Tur36] |
| Monadic first-order logic | NEXPTIME-complete | [BGW93] |
| OWL DL | NEXPTIME-complete | [BCM+03] |
| OWL Lite | EXPTIME-complete | [CGL+05] |
| Propositional Logic | NP-complete | [Coo71] |
| RCC-5 $\mathbb{R}_{32}$ | NP-complete | [JD97] |
| RCC-5 $\mathbb{R}_5^{28}$ | P | [JD97] |
| RCC-5 $\mathbb{B}_5$ | P | [Neb95] |

Table 5.4: Some formal languages and their complexity

## 5.5 Contributions and Future Work

This chapter presented a number of optimizations aimed at reducing the number of proof obligations necessary to calculate a deductive closure on an alignment. The optimizations were shown to have different efficacy depending on the number of constraints placed on the taxonomies. Although the optimizations were somewhat effective in reducing the time necessary to calculate the closure in the large-scale application of CleanTax, the framework could still use some acceleration. One approach for future work is to apply other $\mathcal{L}_{\text{FOL}}$ reasoners, such as Vampire [RV02], and to reduce the overhead created by calling reasoners hundreds of thousands of times. An alternate approach is to explore other subsets of first-order logic. The second half of this chapter covered three possibilities: Description Logics, propositional logic, and the $\mathbb{R}_5^{28}$ subset of RCC-5. Table 5.4 summarizes the languages discussed here, with the computational complexity of those languages.

Finally, the independence of the many reasoning tasks involved in calculating the deductive closure of an alignment lends itself to parallelization on a computer cluster. Future optimizations will aim toward optimal utilization of multiple CPUs.

# Chapter 6

# Merging Taxonomies

## 6.1 Overview and Objectives

This chapter[1] focuses on *merging* multiple taxonomies based on a given alignment. The primary contribution of this chapter is a set of algorithms for merging taxonomies in a way that results in a new, unified taxonomy that maintains links to the original sources. The approach has the following main advantages:

**RCC-Based Articulations.** Unlike the articulation relationships supported in most ontology merging systems [DMQ05, NM03, KV04, KVS06, SM01b], CleanTax uses articulations based on RCC relationships. As already discussed, RCC-based articulations mirror the articulations seen in biological taxonomic alignments [KSBG00, FPW07]. The RCC algebra also supports the representation of incomplete knowledge via explicit disjunctive relationships between taxa.

**Merge Results as Taxonomies.** Because the result of a merge is itself a taxonomy, it is amenable to the application of known taxonomic operations. For example, from a merged taxonomy we can determine if the merge result adheres to specific taxonomic constraints,

[1] Much of this chapter is drawn from [TBL08] and [TBL09b].

if it is logically consistent, if it contains synonyms, if it contains uncertainty that can be reduced, or if it contains redundant articulations.

**Links to Original Sources.** As will be demonstrated in Chapter 7, merged taxonomies that contain links to source taxonomies may be used by applications such as data aggregators that combine observations of species from many data sources (occurrence counts, height and weight measurements, etc.) – where each source may use a different "field guide" (species taxonomy). For example, using a merged taxonomy, it becomes possible to: determine if two datasets contain observations of the same species even when the species are described using different taxonomies; convert datasets into equivalent ones but with a different taxonomy; and discover datasets via concepts drawn from familiar, underlying taxonomies [DMQ05].

**Simplified Taxonomic Views.** A single merged taxonomy can also help users understand the effect of articulations between source taxonomies. Although a large set of articulations might be consistent, it still may be difficult to understand all implications simply by considering pairwise combinations of taxa. Providing a minimal "taxonomic" view of the product of the alignment can help a user understand the impact of an alignment and refine it as necessary.

## 6.2 Related Work

Taxonomies may be seen as simplified ontologies. There has been a considerable amount of work on merging ontologies. Much of this work has focused on using instances [SM01a] or lexical information in the names and definitions of classes [KJH+05] to automatically generate articulations between concepts in separate ontologies. The ontologies are then merged together based on these articulations. The use of instances and lexical information in these systems differs from the work described here, which focuses specifically on the structure of the taxonomies being merged (i.e., the concepts, or taxa, and their relationships). Of

the many tools and approaches for ontology merging, the OntoMerge [DMQ05], Chimæra [MFRW00a, MFRW00b], and iPrompt [NM03] systems are most similar to CleanTax.

In OntoMerge, the merge of two ontologies is the union of the axioms defining the ontologies and the articulations between them. The approach employed by OntoMerge is meant to assist in the translation of data represented using terms from one ontology into data that can be represented using another ontology. In addition to data translation, OntoMerge is meant to support query answering between ontologies, so that queries stated using terms of one ontology may be rewritten into queries over other ontologies. In both of these scenarios, the merge must maintain connections to the source ontologies. Unlike CleanTax, which uses relations drawn from the RCC-5 algebra, articulations in OntoMerge are represented using an enriched full first-order logic (WebPDDL). The result of merging ontologies in OntoMerge is represented as a set of first-order logic formulas, whereas in our approach we always construct a new "unified" taxonomy $T$ having the structure defined above. This taxonomy can further be simplified in our approach, resulting in taxonomic merges that are often more intuitive and easier to understand for end users.

The Chimæra and iPrompt systems differ from OntoMerge in that their goal is primarily to create a new ontology from the source ontologies. Chimæra and iPrompt's merges involve fusing identical terms in the source ontologies into a new term, and determining the subsumption and disjointness relations between the classes in the separate ontologies. Unlike both CleanTax and OntoMerge, these systems are interactive, giving users hints about how concepts in the separate ontologies may relate. Whereas CleanTax restricts articulations to relations covered by the RCC-5 algebra, Chimæra and Prompt use frame-based and Description Logic based representation languages. Finally, unlike OntoMerge and CleanTax, determining the relationships among source concepts from a merged ontology is not supported by Chimæra and iPrompt (although iPrompt does maintain a separate log describing the process used in creating the merged ontology). Maintaining these source relations is critical for applying merged taxonomies, e.g., for data discovery and integration.

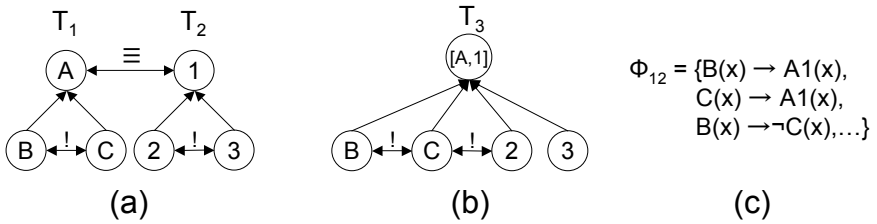



Figure 6.1: Given the alignment in (a), the merge in (b) violates all the described desiderata, except for D5 (closure). The merge in (c) shows a violation of D5.

## 6.3 Desiderata

Listed here are a number of desirable features that systems for creating, using, and managing taxonomy merges should have, along with descriptions of how the features are supported by CleanTax.

The following assumes two taxonomies $T_1$ and $T_2$, and a set $A_{12}$ of articulations between them. As usual, taxonomies and articulations in CleanTax are formalized as sets of first-order formulas. For the taxonomies and articulations defined above, we denote the union of their respective first-order formulas as:

$$\Phi_{12} = \Phi_{T_1} \cup \Phi_{T_2} \cup \Phi_{A_{12}}.$$

We denote the taxonomy $T_3$ resulting from the the merge of $T_1$ and $T_2$ as:

$$T_3 = T_1 \oplus_{A_{12}} T_2.$$

### 6.3.1 Desiderata for Merge Results

The following desiderata focus on desirable features of the output (merge result) of a merge operation.

**(D1) Conservative.** The result of a merge should preserve all consequences of the union of the source taxonomies and articulations. Formally, if $\Phi_{12} \models \varphi$, then $T_3 \models \varphi$. When this is true, we can say the merge result is *conservative*: what was true before is still true—consequences are preserved. For example, the merge of the alignment in Figure 6.1(a) shown in Figure 6.1(b) violates this desiderata because the disjointness between taxa 2 and 3 is not maintained. One ramification of this desiderata is that it places restrictions on merge operations that attempt to simplify the representation of the merge result (i.e., it should still be possible to obtain all consequences of the alignment via the simplified version of the result).

**(D2) Sound.** The result of a merge should not introduce consequences that do not follow from the alignment. We consider two different notions of soundness: soundness and soundness under renaming. In *soundness*, all inferences that follow from the merge result should also be true of the alignment: if $T_3 \models \varphi$ then $\Phi_{12} \models \varphi$. Soundness is violated if the merge result includes new taxa that did not appear in either of the source taxonomies, e.g., if the merge result includes new taxa representing the fusion of equivalent source taxa. On the other hand, *soundness under renaming* is not violated by taxa that have been introduced during the merge if these taxa are equivalent to taxa in the original taxonomies. For example, if the relation $N \supsetneq M'$ (i.e., $N$ is a proper superset of $M'$) is in the merge, where $N$ is a taxon in one taxonomy and $M'$ is a taxon created during the merge, strict soundness will always be violated (because $M'$ is not mentioned in either $T_1$, $T_2$, or $A_{12}$). However, if $M \equiv M'$ where $M$ is in one of the taxonomies, and $N \supsetneq M$ is a relation in the original alignment, then soundness under renaming is not violated. Figure 6.1(b) violates both soundness and soundness under renaming because it introduces disjointness between taxa C and 2 and this disjointness does not follow from the alignment in Figure 6.1(a).

**(D3) Source Maintaining.** Many of the use cases for a taxonomic merge operator require a connection between the merged taxonomy and the source taxonomies. This type

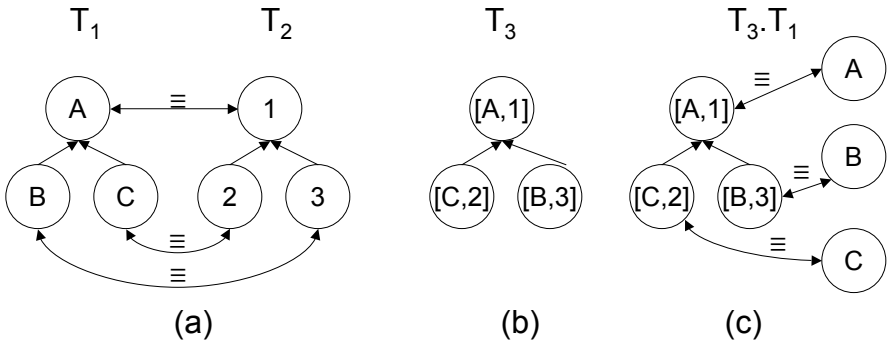



Figure 6.2: Projecting Taxonomy 1 from the Merge.

of connection is required, e.g., to translate datasets from one taxonomy to another. It is also required to query one taxonomy using terms from a second. In both of these cases, without the connection between the merged taxonomy and the sources, there is no way to determine how the terms in the source taxonomies relate to those in the merged taxonomy. Approaches such as OntoMerge [DMQ05] contain these types of connections because the merged ontologies are precisely the formulas derived from the source ontologies and articulations. Alternatively, in approaches such as iPrompt [NM03], these connections are maintained in a more indirect way, e.g., by recording the decisions made during the creation of the merge result, or in a separate mapping file. However, the connections between source taxonomies and the merge result that are maintained using iPrompt's provenance-based mechanism are difficult to exploit in data translation tasks.

To help leverage the applicability of a source-maintaining merge result, we introduce the "source projection" of a merged taxonomy. Given a merged taxonomy $T_3$ derived from taxonomies $T_1$ and $T_2$ and articulations $A_{12}$, the source projection (or simply projection) of the merge result provides linkages to the source taxonomies. For example, Figure 6.2(b) shows a merge of the alignment in Figure 6.2(a). The projection of $T_1$ from $T_3$ in Figure 6.2(c), denoted $T_3.T_1$, shows how the taxa in $T_3$ relate to the taxa in $T_1$. Note that projection does not recreate the entire source taxonomy. It simply provides linkages from the merge to its sources. Figure 6.3 shows how the projection might be used in a merge. Figure 6.3(a) shows three taxonomies and two sets of articulations. After merging $T_1$ and

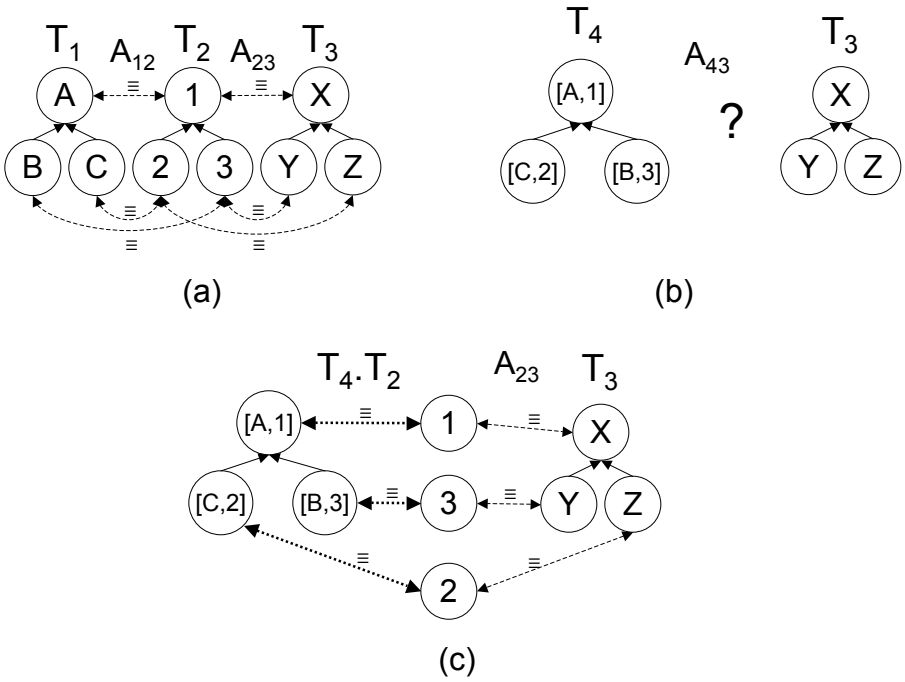



Figure 6.3: Using the Projection.

$T_2$, the resulting taxonomy might look like $T_4$ in Figure 6.3(b). When the merge of $T_4$ is attempted, the articulations in $A_{23}$ cannot apply because of the renamed nodes in $T_4$, and the absence of a known set of articulations between $T_4$ and $T_3$. To resolve this mismatch between the taxa in $T_4$ and those referenced in $A_{23}$, $T_2$ is projected from $T_4$ in Figure 6.3(c), and this projection provides connection points for the $A_{23}$ articulations. More concisely,

$$T_4 = ((T_1 \oplus_{A_{12}} T_2).T_2) \oplus_{A_{23}} T_3$$

.

The merge in Figure 6.1(b) violates the source maintaining desiderata because there is no connection between taxon [A,1] in the merge and either taxon A or 1. In other words, $T_3.T_1$ cannot be calculated.

### 6.3.2 Desiderata for Merge Operations

The following desiderata focus on desirable properties of the merge operation itself:

**(D4) Closed.** The result of a merge operator should be output as a taxonomy. If the result of the merge operation is itself a taxonomy, all of the operations that apply to taxonomies may also be automatically applied to the merge result. These operations include checking the merge result for consistency, displaying the result visually, determining the minimal set of axioms to describe the merge result, and potentially merging the result with additional taxonomies. The set of logic axioms in Figure 6.1(c), though it may represent a merge result that satisfies all other desiderata, is not a taxonomy according to our definition of a taxonomy; it has neither a specified set of taxon names $N$, nor a specified partial order $\preccurlyeq_N$.

**(D5) Associative and Commutative.** Given a sequence of (e.g., binary) merge operations, the order in which the operations are executed should not matter: $(T_1 \oplus_{A_{12}} T_2) \oplus_{A_{23}} T_3 = T_1 \oplus_{A_{12}} (T_2 \oplus_{A_{23}} T_3)$. Besides being more intuitive for users, this desiderata is also important for optimization within systems for managing taxonomies. For example, if $T_2$ and $T_3$ have been merged in the past, and the result is easily retrievable, it would be beneficial to be able to use that cached result when determining $(T_1 \oplus_{A_{12}} T_2) \oplus_{A_{23}} T_3$. The merge result in Figure 6.1(b) loses associativity in a merge like $(T_1 \oplus_{A_{12}} T_2) \oplus_{A_{23}} T_3$ because taxon 1 in $T_2$ no longer exists in the merge result; it is replaced by taxon [A,1]. This replacement of taxon names means the articulations in $A_{23}$ involving taxon 1 from $T_2$ will not apply to the merged taxonomy resulting from $T_1 \oplus_{A_{12}} T_2$, and so will not be reflected in the subsequent merge with $T_3$. Similarly, given two taxonomies, the order in which they are provided in a merge operation should not matter, i.e., commutativity should also hold: $T_1 \oplus_{A_{12}} T_2 = T_2 \oplus_{A_{21}} T_1$.

Finally, it is also desirable for a taxonomy merged with itself to result in the original taxonomy, i.e., idempotence should also hold: $T_1 \oplus_{A_{11}} T_1 = T_1$.

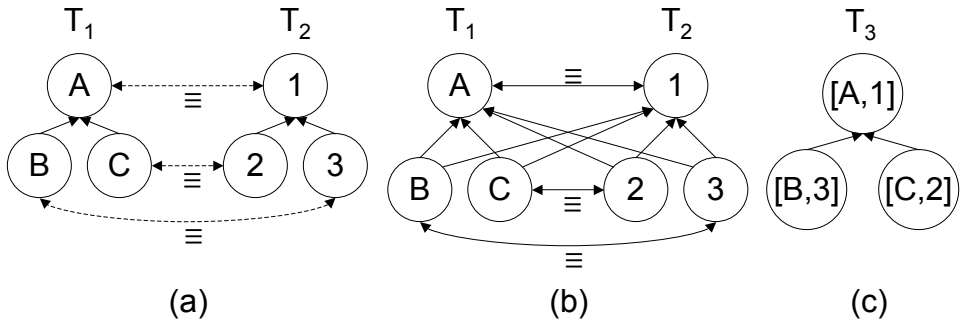



Figure 6.4: Merging with and without fusing equivalent taxa.

**(D6) Minimal.** A taxonomy free of redundant information is often easier to use and understand. For example, the alignment in Figure 6.4(a) could be merged as in Figure 6.4(b); however, this merge contains a great deal of redundant information. Combining equivalent nodes, as in Figure 6.4(c) eliminates the redundant information and creates a merge that is easier to understand.

**(D7) Scalable.** As described above, merge operations should be able to scale-up to large taxonomies, containing many articulations, while preferrably providing reasonable response-time, e.g., for articulation providers so they can quickly see merge results, for systems managing taxonomies, and for systems performing taxonomy-based data discovery, translation, and integration services.

The following section presents a merge algorithm that satisfies each of the above desiderata.

## 6.4 Taxonomy Merging in CleanTax

The merge algorithm begins by using a reasoner to calculate the deductive closure of the union

$$\Phi_{12} = \Phi_{T_1} \cup \Phi_{T_2} \cup \Phi_{A_{12}}$$

of the logic axioms describing the source taxonomies and the articulations. This type of merge is much like that described in the OntoMerge system [DMQ05], whose merge result is represented by the set of logic statements rather than as a new taxonomy (violating the closure requirement of Section 6.3).

CleanTax constructs a taxonomic merge by coercing $\Phi_{12}$ into the signature for a taxonomy $T = (N, \preccurlyeq_N, \Phi)$. This step consists of determining the taxa involved in the merged taxonomy, deriving the transitive reduction of the partial order describing the relationships between those nodes, and deriving the additional taxonomic constraints.

We determine $N, \preccurlyeq$, and $\Phi$ initially as follows. $N$ is simply the set of taxa that appear in the initial taxonomies. The transitive reduction $\preccurlyeq$ is determined by constructing a graph of the taxa in $N$ where each taxon is a node, and there is a directed edge between any two taxa $N_1$ and $N_2$ when the $\mathbb{R}_{32}$ relation $\subsetneq$ or $\{\equiv, \subsetneq\}$ can be deduced from the deductive closure. Once this graph has been constructed, the transitive reduction may be determined using a standard transitive reduction algorithm [AGU72]. Finally, $\Phi$ is simply the union $\Phi_{T_1} \cup \Phi_{T_2} \cup \Phi_{A_{12}}$.

Once this initial taxonomy is formed, the final merge is created by merging taxa found to be equivalent, due to provided or inferred articulations.

We define an equivalence relation on $N$ such that:

$$a \sim b \quad \text{if} \quad \Phi \models \forall x : a(x) \leftrightarrow b(x),$$

where the equivalence class of $a \in N$ is $[a] = \{x \in N \mid x \sim a\}$. We say that taxonomy $T$ has *synonyms* if for some $a, b \in N$ with $a \neq b$ we have that $a \sim b$; otherwise $T$ is called *synonym-free*. Using this definition we can construct a unique, synonym-free version of the

initial merge result. We call this simplified version a *quotient taxonomy* $T_{/\sim}$ such that:

$$\begin{aligned} N_{/\sim} &= \{[a] \mid a \in N\}, \\ \preccurlyeq_{/\sim} &= \{([a],[b]) \mid [a] \preccurlyeq [b] \text{ if } a \preccurlyeq b\}, \\ \Phi_{/\sim} &= \{[\varphi] \mid \varphi \in \Phi\}. \end{aligned}$$

Here for every FO formula $\varphi$, we define its quotient $[\varphi]$ to be the formula where each atom a(x) has been replaced by the atom [a](x).

We briefly describe how the above merge algorithm satisfies the desiderata of Section 6.3. First, based on the deductive closure, the results produced by the merge operation are conservative and sound under renaming. Namely, all consequences of the union of the source taxonomies and articulations are preserved, and no new information has been added to the merge result that could not be derived from the original taxonomies and articulations, where each taxon in $N_{/\sim}$ is equivalent to at least one source taxon. Merge results are also source maintaining. In a quotient taxonomy, each taxon $[a] = \{x_1, x_2, \dots\}$ for $a \in N_{/\sim}$ implicitly carries its linkages to corresponding source taxa, where the source projection operation simply selects the desired source taxa of $[a]$. For instance, for the [A,1] taxon in Figure 6.4(c), $N = [T_1.\text{A}, T_2.1]$ such that the source projection $T_3.T_1$ is $\{(T_3.\text{A1}, T_1.\text{A}), (T_3.\text{B3}, T_1.\text{B}), (T_3.\text{C2}, T_1.\text{C})\}$. This projection can then be either rendered into a set of first-order axioms, or can be used to rewrite a set of articulations. In the former case, each pair in the projection $(m, n)$ would add an axiom $\forall x.m(x) \leftrightarrow n(x)$ to $\Phi$. In this case we can define $(T_1 \oplus_{A_{12}} T_2).T_2 = (N_{T_3} \cup N_{T_2}, \preccurlyeq_{T_3}, \Phi_{T_3} \cup \Phi_{T_3.T_2})$. In the latter case, each taxon in the set of articulations matching the second element of a pair in the projection will be replaced with the name of the first element of that pair.

Furthermore, the merge operation itself is closed since it results in a taxonomy $T$ as defined above. The merge is also commutative since it is possible to invert a set of articulations, and similarly associative under source projection. For quotient taxonomies, the merge operation is idempotent. That is, given two identical quotient taxonomies the same

quotient taxonomy is returned.[2] Quotient taxonomies can be considered minimal views being synonym-free and consisting of the transitive reduction. And finally, as we describe further in the following section, the merge operation can scale-up to large taxonomies, in part due to CleanTax's use of RCC constraints.

## 6.5 Experiments and Discussion

The merge algorithm has been implemented within the CleanTax system and tested using the largest alignment in the Ranunculus dataset: one taxonomy covering 218 taxa, the other covering 142 taxa, with 206 articulations between them. Each taxonomy is three levels deep covering the genus, species, and variety biological ranks.

The first step in creating the merge is translating the taxonomies and articulations into monadic first-order logic and determining all the relationships implied by the resulting axioms. Once these calculations have been made, the merge is computed very quickly. The limiting factor of the algorithm is the calculation of the transitive reduction, for which we used the **tred** filter that comes with the graphviz software package[3]. **Tred** uses a depth-first search algorithm of complexity $O(V * E)$ [AGU72, IR88]. In the current context, $V$ is the number of taxa and $E$ is the number of articulations describing inclusion maximally informed relations (either $N \subsetneq M$ or $N\{\equiv, \subsetneq\}M$). The other steps of the algorithm are $O(E)$ where $E$ is the number of `mir` articulations. On average (after 5 runs with little variance between them) merging the two taxonomies described above took 84 milliseconds, 62% of which was spent determining the transitive reduction.

A primary advantage of the CleanTax framework is the ability to apply a variety of taxonomic constraints when reasoning and merging across taxonomies. Calculating the merge for the Ranunculus sub-taxonomies, which contained on average 8 taxa each, took on average 18 milliseconds, 99% of which was spent determining the transitive reduction.

Figure 6.5 shows the impact of the constraints on the merge of one of these sub-

[2] Note that it is also straightforward to convert source taxonomies into corresponding quotient taxonomies.
[3] http://www.research.att.com/sw/tools/graphviz/

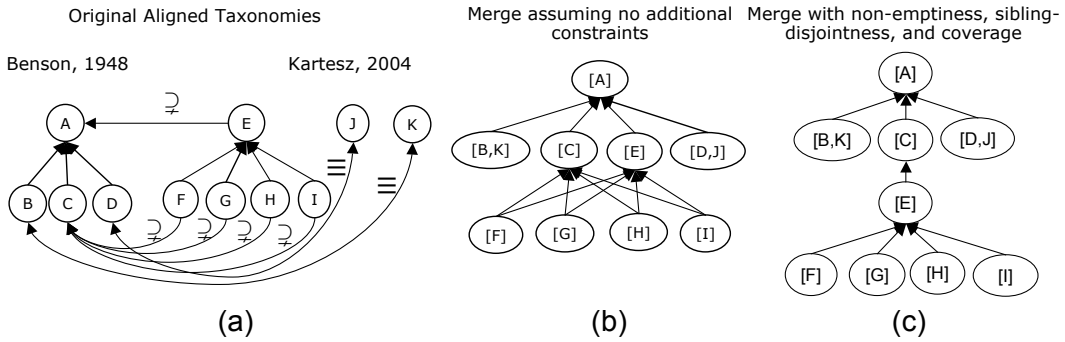



Figure 6.5: Merging *Ranunculus hispidus* under different assumptions. For clarity, the disjointness relations between taxa in (c) are not shown. See text for further detail.

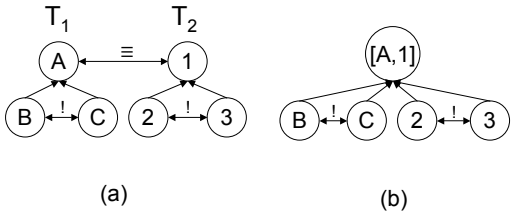



Figure 6.6: Constraints placed on taxonomies before the merge may not apply to the result of the merge.

taxonomies. The two sub-taxonomies for the species *Ranunculus hispidis* and their articulations are shown in Figure 6.5(a). When no additional assumptions are made, the merge results in Figure 6.5(b). It is important to recognize that in Figure 6.5(b), the lack of an edge between two taxa represents the situation where either a transitive edge has been removed in the transitive reduction or nothing is known about the relationship between the taxa. Thus, in Figure 6.5(b) the relationship between taxa [C] and [E] is completely unknown. Applying the non-emptiness constraint to all the taxa in the taxonomies results in the additional knowledge that taxa [C] and [E] are not disjoint.

Figure 6.5(c) represents the merge when the sibling disjointness, coverage, and non-emptiness constraints are assumed. In this merge, the taxon labeled [E] becomes a child of [C]. For clarity, the many disjointness relations between taxa in Figure 6.5(c) are not shown: the taxa [F], [G], [H], and [I] are mutually disjoint, the taxa [B,K], [C], and [D,J] are mutually disjoint, and each child of [E] is disjoint from [B,K] and [D,J].

It is important to note that when GTCs are applied to the taxonomies being merged, they are not automatically applied to the result of the merge. For example, in Figure 6.6, although the two taxonomies shown in (a) both exhibit the sibling disjointness constraint, the resulting merge in (b) does not; nothing is known about the relationship between taxa C and 2, for instance. Applying the sibling disjointness constraint to the merged result would be adding additional information, violating the soundness desideratum. If the articulation provider expects taxa C and 2 to be disjoint, this articulation must be added to the alignment in Figure 6.6(a).

## 6.6 Comparison to Related Systems

Fundamental differences between the OntoMerge, iPrompt, Chimæra, and CleanTax approaches complicate comparisons between the systems. For example, OntoMerge does not have an explicit merge phase to compare with the CleanTax merge. iPrompt and Chimæra are interactive systems in which users merge ontologies by iteratively creating articulations and performing the merge, whereas the CleanTax merge assumes a set of articulations has been provided and performs the merge in one step. In all cases, the languages used for representing articulations differ, and the types of reasoning applied differ. Table 6.1 details some of these differences between the systems.

| | **CleanTax** | **OntoMerge** | **Chimæra** | **iPrompt** |
|---|---|---|---|---|
| **When Merge Happens** | After articulations and reasoning | No real merge | During articulation | During articulation |
| **Disjunctive Relation Support** | Yes | No | No | No |
| **Types of Reasoning Supported** | Monadic FOL RCC | Forward and backward chaining | Extended FOL | Description logic |
| **Result of Merge** | Taxonomy | Knowledge base | Taxonomy | Ontology |
| **Support for Roles and Union** | No | Yes | Yes | Yes |

Table 6.1: Comparing CleanTax to OntoMerge, Chimæra, and iPrompt .

The differences in how and when the systems apply reasoning during the merge operation impacts the result of their merge operation. Whereas CleanTax performs a deductive

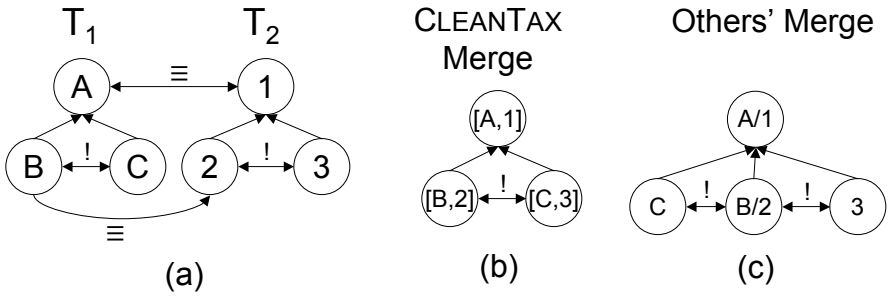



Figure 6.7: Comparing merges for the taxonomies in (a) under the parent-coverage constraint. CLEANTAX correctly merges taxa C and 3 (b) while the others do not (c).

closure before the merge occurs, neither iPrompt nor Chimæra appear to do so (although they easily could). And while OntoMerge supports an extended full first-order logic, its reasoning over that logic is quite restricted. An effect of these differences in reasoning may be seen in the results of the merge in Figure 6.7. When the sibling disjointness, coverage and non-emptiness GTCs are in place, CLEANTAX merges taxa C and 3. Chimæra, iPrompt, and OntoMerge each leave 3 and C distinct.

## 6.7 Conclusion

This chapter has presented a formal approach for merging taxonomies within the CLEANTAX system. This work is motivated by current problems in managing, integrating, and exploiting large biological classifications including species taxonomies. As such, the chapter has also identified a number of requirements related to merging taxonomies, and described how the proposed merge approach satisfies them. Experimental results of an implementation of the merge approach have been provided, using a number of real-world species taxonomies and articulations created by a domain expert. The results suggest that the merge approach is well suited for handling large taxonomies and complex sets of articulations.

# Chapter 7

# Merging Taxonomically Classified Data

## 7.1 Introduction

This chapter[1] addresses the problem of merging datasets when the domains of the data attributes overlap but are not equivalent. Consider, e.g., two datasets that report observations of the presence or absence of biological taxa in a given region and at a given time.[2] Each of the dimensions, biological, spatial, and temporal, may be represented using a taxonomy, and the datasets may each use different taxonomies for any given dimension. In the absence of any information about the relationship between the concepts in their taxonomies, the datasets can be naively merged by simply concatenating the observations into a single dataset. This method, however, may result in a self-contradictory dataset, or one that contains hidden redundancies and uncertainty. Given an alignment between two taxonomies, the datasets can be merged in a more informed way. This chapter presents a methodology for merging datasets that takes advantage of alignments between taxonomies

[1] Much of this chapter is drawn from [TBL09b] and [TBL10].

[2] Presence datasets such as this are very common. For example, epidemiological studies track the presence of diseases over time and space [CHSR09]. In ecological and biodiversity research, many datasets stored in data repositories (such as Metacat [BJBM01]) are composed of lists of biological taxa found in specified geographic extents over given periods of time.

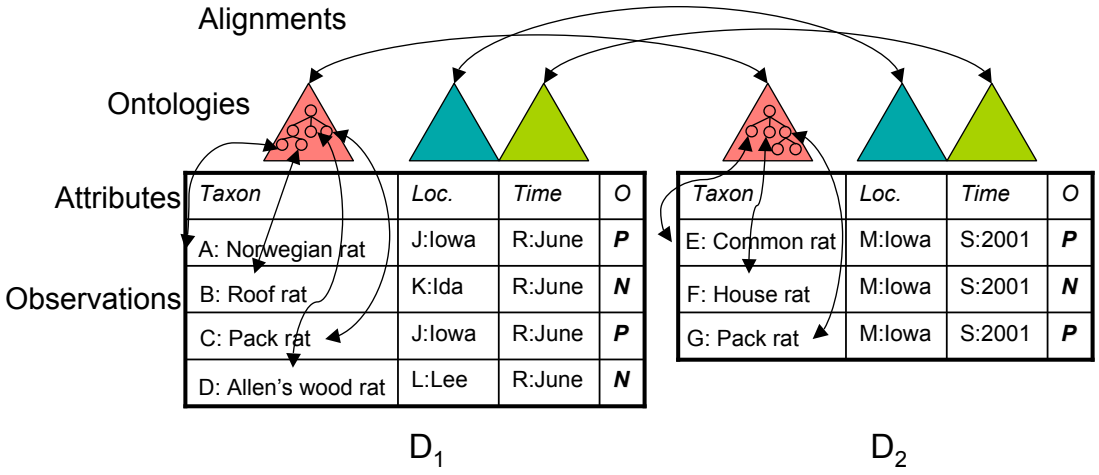



Figure 7.1: Two datasets, with corresponding ontologies and ontology alignments.

while detecting contradictions, and minimizes uncertainties that may arise in the merge.

Figure 7.1 presents a simple example involving two presence datasets $D_1$ and $D_2$ that describe types of rats found to be present or absent at specific places and times. The *Taxon* column represents biological taxa, preceded by an abbreviation (e.g., "A" for Norwegian rat). The taxonomies used to define and relate the taxa are represented by ontologies depicted above the Taxon columns of the datasets. The creators of the two datasets may have used different field guides to identify the taxa, in which case the Taxon ontologies must be aligned to account for differences between the field guides. The *Loc* column represents spatial locations: counties in Iowa in the first dataset and the State of Iowa in the second dataset. The ontologies from which the location names are drawn are represented above their respective columns, and an alignment relates the location names used in the datasets. Note that Iowa is both the name of a US State and of a county in that state. *Time* records when the observations are made. Finally, *O* records whether or not a given taxon, at a given place and a given time, is present (`P`) or not present (`N`) (absent). The presence of a taxon does not imply that only *one* instance of that taxon was seen at that place, at that time. In addition, presence and absence are modeled as complements; a taxon cannot be both present and absent at a given location and time.

**Merge Scenarios.** Each dataset shown in Figure 7.1 provides a perspective on the state of the world at a given place and time, according to a given observer. We call each dataset a *scenario*. Merging the datasets should provide a more complete description of the state of the world. However, it may not be clear how to best merge the datasets, and many scenarios may be possible. For example, the merged dataset shown in Table 7.1(a) describes the scenario arising from a simple union of the source datasets. Although it seems like an obvious merge, it makes many, possibly incorrect, assumptions. First, it assumes every name is distinct from every other name. However, concepts between datasets can be equivalent, potentially rendering the merge in Table 7.1(a) inconsistent. If concept $A$ in $D_1$ (Norwegian rat) is equivalent to concept $F$ in $D_2$ (House rat), and $R$ in $D_1$ is equivalent to $S$ in $D_2$ (both studies were carried out in June, 2001), and concept $J$ in $D_1$ (Iowa County) is a proper part of $M$ (Iowa State) in $D_2$, then the observations corresponding to rows 1 and 6 in Table 7.1(a) would be reporting both the presence and absence of the same taxon at the same place and time. Table 7.1(a) further assumes that an unreported taxon does not imply the absence of that taxon. If an unreported taxon is assumed to be absent, and, e.g., if Norwegian rat in $D_1$ is disjoint from all the taxa listed in $D_2$, it would be problematic that $D_1$'s observer reported the presence of at least one Norwegian rat and $D_2$'s observer did not.

Table 7.1(b) and (c) present two alternative scenarios. Table 7.1(b) assumes an alignment in which certain concepts are equivalent (e.g., $A \equiv E$ as represented by the new taxon $AE$). The alignment also asserts that certain concepts are proper parts of others. For example, concept $J$ is aligned as a proper part of concept $M$ ($J \subsetneq M$). This is represented by introducing new location concepts: $JM$ represents the region where $J$ and $M$ overlap ($J \cap M$), and $\bar{J}\bar{K}\bar{L}M$ represents the region of $M$ that excludes $J, K$, and $L$ ($M \setminus (J \cup K \cup L)$).

**Sources of Uncertainty.** Uncertainty induces multiple possible merges. For example, the different merges in Table 7.1 occur because of uncertainty in the alignment between ontologies: the concepts A and E might be distinct concepts, as in Table 7.1(a), or equiva-

| *Taxon* | *Loc.* | *Time* | *O* |
|---|---|---|---|
| A | J | R | P |
| B | K | R | N |
| C | J | R | P |
| D | L | R | N |
| E | M | S | P |
| F | M | S | N |
| G | M | S | P |

(a)

| *Taxon* | *Loc.* | *Time* | *O* |
|---|---|---|---|
| AE | JM | RS | P |
| AE | $\bar{J}\bar{K}\bar{L}$M | RS | P |
| BF | KM | RS | N |
| BF | $\bar{J}\bar{K}\bar{L}$M | RS | N |
| CG | JM | RS | P |
| CG | $\bar{J}\bar{K}\bar{L}$M | RS | N |
| D | LM | RS | N |
| D | $\bar{J}\bar{K}\bar{L}$M | RS | N |

(b)

| *Taxon* | *Loc.* | *Time* | *O* |
|---|---|---|---|
| *AE* | *JKLM* | *RS* | P |
| *BF* | *JKLM* | *RS* | N |
| *CG* | *JKLM* | *RS* | P |
| *D* | *JKLM* | *RS* | N |

(c)

Table 7.1: Three possible merges of the datasets in Figure 7.1.

lent concepts, as in Table 7.1(b). This uncertainty may have been explicitly stated by the ontology aligner ($A \equiv E$ or $A \,!\, E$), or it may have been inferred from an incomplete alignment. This kind of uncertainty is called *disjunctive relation uncertainty* (DRU) because it involves a disjunction of relations (equivalent *or* disjoint, in this case). Disjunctive relations may also exist within individual ontologies. For example, the traditional interpretation of "isa" as "equals or is included in" [Bra83] is a disjunctive relation.

Even when the relationship between two concepts is certain, the relationship may lead to uncertainty. For example, if an alignment holds that concept $A$ according to $D_1$ is a kind (i.e., proper subset) of concept $E$ according to $D_2$ ($A \subsetneq E$), it is unclear whether or not any of the $E$'s reported in dataset 2 are also $A$'s. There are two possibilities: either all the observed rats are both $A$'s and $E$'s ($AE$), or some of the rats are $E$'s but not $A$'s ($\bar{A}E$). This source of uncertainty is called *basic relation uncertainty* (BRU) because it arises from basic set relations. Whereas disjunctive relation uncertainty exists at the ontology level, basic relation ($\mathbb{B}_5$) uncertainty occurs at the level of the observations in the datasets. To reliably resolve this uncertainty, one would have to ask for clarification from the dataset's

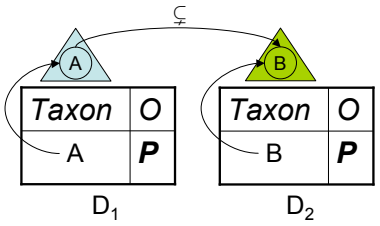
observer.

The goal is to create dataset merges free of BRU and DRU. While BRU and DRU may appear in source datasets, in general high quality datasets do not contain these types of uncertainty and many "best practices" guides for data collection [Cha05] specifically recommend avoiding these types of uncertainty. This chapter provides algorithms for merging datasets that are free of BRU and DRU, as well as those that are not. However, the algorithm for merging datasets that do not contain BRU or DRU is considerably more efficient than the one for merging datasets that already contain uncertainty.

**Contributions and Road Map.** This chapter contributes a novel modeling framework for merging datasets with aligned domains under uncertainty. It describes several sources of uncertainty within data sets as well as arising from the merging of datasets, and presents a possible worlds semantics for managing this uncertainty. Finally, the chapter provides algorithms for merging datasets in this context, providing NEXP-time algorithms for the general case of generating possible worlds, and an NP-time SAT-based solution for the common case of merging source datasets that do not contain BRU and DRU.

## 7.2 Basic Approach

This section provides an informal description of the elements involved in dataset merging, and a high-level description of the approach. Datasets are defined as relations over finite sets of attributes. Data items within a data set are tuples of values, where the values are drawn from their respective attribute domains. The values represent *concepts* (classes), which are sets of instances. For example, taxa are sets of (perhaps unknown) biological specimens, locations are sets of points in space, and times are sets of moments. The attribute domains may be structured, containing the domain concepts and relations between them stated in some language (e.g., first-order logic, monadic logic, or Description Logic). They are called *ontologies* and a dataset's collection of ontologies is called its *metadata*. Internally




(a)

| World | $AB$ | $A\bar{B}$ | $\bar{A}B$ | $\bar{A}\bar{B}$ |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 0 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 16 | 0 | 0 | 0 | 0 |

(b)

| World | $AB$ | $\bar{A}B$ |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 0 |

(c)

| *Taxon* | *O* |
|---|---|
| $AB$ | P |
| $\bar{A}B$ | P |

(d)

| *Taxon* | *O* |
|---|---|
| $AB$ | P |
| $\bar{A}B$ | N |

(e)

Figure 7.2: (a) A very simple scenario, (b) its initial world set, (c) the reduced possible world set, (d) and (e) the corresponding merged datasets.

consistent source datasets are assumed. Inconsistency, however, can occur in a number of places. A dataset may contain contradictory information if, e.g., it states both the absence and presence of a taxon at a given place and time. A dataset may also be inconsistent with its metadata, e.g., if the metadata states that taxa $A$ and $B$ are equivalent (represent equivalent sets), but the data say that $A$ is present at a given place and time and $B$ is absent. Finally, the ontologies in the metadata may be inconsistent. A *legal* dataset is one that does not violate any of these consistency constraints. An *unambiguous* dataset is a legal dataset that contains neither basic nor disjunctive relation uncertainty.

Each scenario in Table 7.1 describes one unambiguous dataset. Each possible merged dataset is one of many possible worlds [LL59, AKG87] in a *possible worlds set* (PWS). Given two datasets, one could generate the appropriate PWS by generating an *initial world set* (IWS) containing every conceivable world (restricted by the finite domains of the metadata), including those worlds that violate the alignment and certainty constraints, and then reducing this set by eliminating columns and rows that violate the constraints.

Unfortunately, this approach is intractable. Consider the extremely simple scenario shown in Figure 7.2(a) having two datasets $D_1$ and $D_2$ with taxon $A$ present in $D_1$, and $B$ present in $D_2$. Each dataset has a single biological attribute, and that attribute can only take one value: $A$ for $D_1$ and $B$ for $D_2$, and an articulation between these concepts states

that $A \subsetneq B$. To generate an IWS, first determine all conceivable conditions that may or may not hold based on the concepts in the dataset ontologies. There are four ways to combine the biological concepts $A$ and $B$: a biological specimen might be an example of $AB, A\bar{B}, \bar{A}B$, or $\bar{A}\bar{B}$. Each of these combinations is called a *combined concept*. Each combined concept represents a set of instances, and a dataset reports whether there are no instances of the set present within the context of the dataset (absence), or at least one instance from the set present (presence). The resulting IWS has $2^2 = 4$ combined concepts and $2^4 = 16$ worlds. This IWS can be conveniently represented with a *world set relation* [AJKO08] as shown in Figure 7.2(b). In this table, the conditions are represented as columns, and each world is a row. The number 1 indicates that instances of the combined concept are present in a given possible world, and 0 represents the absence of instances of that combined concept. The first world represents the (impossible) situation in which instances of all the combined concepts are present. This is impossible because the first combined concept $A\bar{B}$ cannot be present (in fact, is not satisfiable) because $A \subsetneq B$.

Once the IWS table is created, it may be reduced by removing combined concepts and possible worlds that violate constraints or are unsupported by the input datasets. For example, because $A \subsetneq B$, $A\bar{B}$ is an impossible combined concept, any possible world involving it should be removed. Similarly, because $D_1$ reported the presence of $A$, and $A \subsetneq B$, $AB$ must be 1 in every possible world, and any world with 0 in that column should be removed. In addition, combined concepts for which there is no evidence should be removed. In this example, the last combined concept of the IWS should be removed because neither dataset describes specimens that are neither $A$ nor $B$. Finally redundant rows created by the deletion of combined contexts should be removed. Removing all of the impossible, redundant, and unsupported information results in the two possible worlds in the PWS shown in Figure 7.2(c). The two merged datasets that correspond to these possible worlds are shown in Figures 7.2(d) and 7.2(e).

In more typical situations, this approach will not work. For example, merging two datasets with three attributes, where each attribute has a corresponding ontology ($O_1, O_2$,

and $O_3$) with $|O_n|$ concepts will result in a IWS with $C = 2^{|O_1|+|O_2|+|O_3|}$ columns, and $2^C$ rows. The simple scenario in Figure 7.1 would lead to an IWS with $2^{7+4+2} = 8192$ combined concepts and $2^{8192}$ worlds; a number of worlds too large to enumerate, much less manipulate. A primary contribution of this work is a set of more tractable algorithms for generating the appropriate PWS.

## 7.3 Framework

**Dimensions, Concepts, and Ontologies.** Distinct types of objects are organized using *classification dimensions* (or *dimensions* for short). This chapter is primarily concerned with three dimensions: *spatial* (e.g., locations and regions), *temporal* (e.g., points in time and intervals), and *biological* (e.g., organisms classified via biological taxonomies). Vocabularies for classifying objects are represented using *ontologies* $\mathbf{O} = (\{C_1, ..., C_n\}, \Sigma)$ each consisting of a finite set of *concepts* and a set of constraints $\Sigma$ on those concepts. Each concept $C$ specifies a set of objects that are considered to be *instances* of $C$. Each ontology $\mathbf{O}$ is associated with a dimension given by a function $dim(\mathbf{O})$. Thus, each concept of a particular ontology classifies objects of the *same* dimension.[3] Biological ontology concepts describe sets of organisms, spatial ontology concepts describe sets of points in space, and temporal ontology concepts represent sets of moments in time.

**Data Sets and Observations.** Datasets are represented as relations $D$ over the schema

$$\mathbf{C}_1 \times \cdots \times \mathbf{C}_n \times \mathbf{D}_1 \times \cdots \times \mathbf{D}_m$$

where each $\mathbf{C}_i$ denotes a *context attribute* and each $\mathbf{D}_j$ denotes a *data attribute*. A total function, $m : C \rightarrow O$, maps each context attribute to an associated ontology, and the domain of the attribute is restricted to the concepts in the associated ontology. A data

[3] An ontology typically contains terms from different dimensions and can be viewed in our framework as consisting of one or more domains.

attribute represents a set of possible values corresponding to observations made over the given context attributes. The example describes the following special case

$$\mathbf{C_B} \times \mathbf{C_S} \times \mathbf{C_T} \times \mathbf{D_O}$$

where $\mathbf{C_B}$ represents a required biological context attribute (e.g., organisms classified via biological taxonomies), $\mathbf{C_S}$ represents an optional *spatial* context attribute, $\mathbf{C_T}$ represents an optional *temporal* context attribute, and $\mathbf{D_O}$ represents a simple data attribute denoting a presence or absence observation over context attributes. In general, presence datasets are represented by one or more records of the form $D(b, s, t, o)$ where $b \in \mathbf{C_B}$, $s \in \mathbf{C_S}$, and $t \in \mathbf{C_T}$ are concepts, and $o \in \mathbf{D_O}$ is either `P`, meaning at least one $b$ was observed in region $s$ during time $t$, or `N`, meaning no instances of $b$ were found in region $s$ during time $t$. Each record in a dataset is an *observation*. Although biodiversity datasets often contain additional context information and measurements [BMS08], the features described above are sufficient to demonstrate the core issues of dataset merging addressed here.

**Absence Closure.** The datasets described so far contained presence and absence information explicitly. In some cases, a dataset may contain only presence information, but intend that absence is implied when an observation is not made. A presence dataset is *closed under absence* if for each context term $b_i \in \mathbf{C_B}$, $s_j \in \mathbf{C_S}$, and $t_k \in \mathbf{C_T}$ there is a record $D(b_i, s_j, t_k, o)$. If no such record exists in the dataset, the dataset may be closed by asserting an absence observation via the record $R(b_i, s_j, t_k, \mathtt{N})$.

**Relationships Between Ontologies.** This work describes merging two datasets of the aforementioned schema. Although the schemas are the same, the ontologies for the biological, spatial, and temporal context attributes may differ between datasets. Concepts within and across ontologies of the same dimension are related through sets of (first-order) constraints $\Sigma$. Given an ontology $\mathbf{O}$, $\Sigma_{\mathbf{O}}$ denotes the constraints of $\mathbf{O}$. As usual, constraints between concepts of different ontologies are called articulations, and a set of ontologies with

articulations between them is called an alignment. This work only considers articulations between concepts that appear in ontologies of the same dimension, $dim(A.1) = dim(A.2)$. A set of alignments, $\mathcal{A} = \{A_1, \cdots, A_n\}$ where $\forall x, y \in \mathcal{A} : x \neq y \rightarrow dim(x.1) \neq dim(y.1)$, is called an *alignment set*.

**Merging Ontologies.** Merging the context ontologies described here is a straightforward generalization of the merge in Chapter 6 which described a method for merging taxonomies under RCC-5 articulations. Given two ontologies $O_1$ and $O_2$ and an alignment $\Sigma_{\mathbf{O_1O_2}}$ describing the RCC-5 articulations between the concepts in $O_1$ and $O_2$, the merge algorithm begins by converting the ontologies to axioms in a first-order language ($\Phi_{O_1}, \Phi_{O_2}$, and $\Phi_{O_1,O_2}$) and using a reasoner to calculate the RCC-5 closure of the union

$$\Phi_M = \Phi_{O_1} \cup \Phi_{O_2} \cup \Phi_{O_1,O_2}$$

of the logic axioms describing the source ontologies and the articulations.

A merged ontology is then created by defining, if necessary, a new concept for each class of equivalent concepts, and rewriting the articulations determined by the RCC-5 closure with the new concept terms. $C_M$ represents the set of predicate names in $\Phi_M$. An equivalence relation on $C_M$ is defined such that:

$$a \sim b \;\text{ if }\; \Phi \models \forall x.a(x) \leftrightarrow b(x),$$

where the equivalence class of $a \in C$ is $[a] = \{x \in C \mid x \sim a\}$. Ontology $O$ has *synonyms* if for some $a, b \in C$ with $a \neq b$, $a \sim b$; otherwise $O$ is called *synonym-free*. Using this definition we can construct a unique, synonym-free version of the initial merged ontology. This simplified version is called a *quotient ontology* $O_{/\sim}$ such that:

$$\begin{aligned} C_{/\sim} &= \{[a] \mid a \in C\}, \\ \Phi_{/\sim} &= \{[\varphi] \mid \varphi \in \Phi\}. \end{aligned}$$

Here for every FO formula $\varphi$, define its quotient $[\varphi]$ to be the formula where each atom a(x) has been replaced by the atom [a](x).

**Data Set Merge Result and World Sets.** The result of merging two datasets $M = Merge(D_1, D_2, \mathcal{A})$ is often a set of possible worlds. Each world represents an unambiguous dataset that has as its metadata the merge of the source datasets' ontologies, and furthermore *respects* the observations in the source datasets. One dataset $D_1$ respects the observations of another $D_2$ ($D_1 \prec D_2$) if $D_1 \models D_2$. For example, a dataset derived from a possible world $D_M$ respects the observations of one of its sources $D_S$ if for every tuple $t$ in $D_S$, $D_M \models t$.

The main challenge addressed in this chapter is (efficiently) determining the possible worlds. Once they have been found, the worlds can be conveniently represented using a single relation $W$ [AKO07]. Start with a set of possible worlds $P$, where each world $p \in P$ is an instance of a relation following the D(b,s,t,o) schema, where $|p|$ is the number of tuples in $p$. For each tuple in each possible world, apply a function $f()$ to create a symbol representing the concatenation of the context attributes. For example, for the tuple $D(b_1, s_1, t_1, \texttt{P})$, create a symbol $b_1 s_1 t_1$. $T$ is the set of such symbols. The attributes of the schema of $W$ are the symbols in $T$, and its arity is $|T|$. Each attribute in $W$ is indexed with values $1 \leq i \leq |T|$.

The tuples in $W$ are created as follows: For a given world $p \in P$ with tuples $\{t_1, ...., t_n\}$, let $t_p$ be a tuple following the schema of $W$ where for $1 \leq i \leq |T|, t_p(W_i) = 1$ if $\exists x \in p$ such that $f(x) = W_i$ and $o(x) = \texttt{P}$; $t_p(W_i) = 0$ if $\exists x \in p$ such that $f(x) = W_i$; and $o(x) = \texttt{N}$ and $p(W_i) = \perp$ otherwise.

**Translation into Logic.** To determine whether or not two datasets may be merged, to ensure the consistency of datasets, and to validate the result of the merge requires reasoning about the datasets, their ontologies, and the relationships between the ontologies. To provide this reasoning, we translate each of these elements into sets of first-order logic

| | |
|---|---|
| $\equiv$: $\forall x : A(x) \leftrightarrow B(x)$. | !: $\forall x : A(x) \rightarrow \neg B(x)$. |
| $\subsetneq$: $\forall x : A(x) \rightarrow B(x)$. | $\oplus$: $\exists x : A(x) \rightarrow ((A(x) \wedge B(x)) \mid (A(x) \wedge \neg B(x)))$. |
| $\supsetneq$: $\forall x : B(x) \rightarrow A(x)$. | $\exists x : B(x) \rightarrow ((A(x) \wedge B(x)) \mid (\neg A(x) \wedge B(x)))$. |

Table 7.2: A monadic logic encoding of articulations of the form $A \circ B$ where $\circ \in \{\equiv, \subsetneq, \supsetneq, \oplus, !\}$. This encoding applies when translating datasets into logic. When translating ontologies and articulations into logic for the purpose of checking their consistency or merging the ontologies, use the encoding in Chapter 4.

formulas.

Each record of a dataset $D$ induces a first-order logic formula as follows: A presence observation denoted by a record of the form $D(b, s, t, \mathtt{P})$ is represented by a formula

$$(\exists xyz) \quad b(x) \wedge s(y) \wedge t(z) \wedge \mathit{present}(x, y, z)$$

where the relation $\mathit{present}(x, y, z)$ holds whenever the biological entity $x$ was present at location $y$ and time $z$.[4] The formula above states that a biological organism $x$ of type $b$ was observed within location $y$ of type $s$ and at time $z$ of type $t$. Similarly, an absence observation denoted by a record $D(b, s, t, \mathtt{N})$ is represented by a formula

$$(\forall xyz) \quad b(x) \wedge s(y) \wedge t(z) \rightarrow \neg \mathit{present}(x, y, z)$$

stating that for each biological entity $x$ of type $b$, location $y$ of type $s$, and time $z$ of type $t$, $x$ was not found within location $y$ at time $z$. Note that this encoding of absence asserts the complete absence of entities of the given biological type throughout the given spatial and temporal contexts. The set of axioms reflecting the observations of a dataset is called $\Phi_{DI}$.

The constraints over the concepts in the ontologies are encoded using monadic logic. More specifically, the ontology constraints in $\Sigma_O$ are restricted to relations from the RCC-5 algebra, plus an additional type of constraint called *coverage*. The coverage constraint states that one concept can be defined as the union of a set of concepts (e.g., $(\forall x) \quad P(x) \leftrightarrow$

[4] Where the formula includes the $S$ and $T$ terms only if these are part of the presence-absence schema.

$C_1(x) \vee \cdots \vee C_n(x)$.) We define $\Phi_O$ as the combined set of formulas generated by translating the RCC-5 constraints in $\Sigma_O$ into monadic logic using the rules in Table 7.2, plus additional coverage constraints. The RCC-5 based articulations between ontology concepts are also represented as monadic logic formulas $\Phi_A$.

A complete dataset, then, is defined as

$$\Phi_{DS} = \Phi_{DI} \cup \Phi_{O_1} \cup \cdots \cup \Phi_{O_n}$$

where $n$ ranges over the ontologies referenced by the dataset.

**Merge-Compatible Data Sets.** To determine whether or not two datasets may be merged, calculate the absence closure for each dataset, if required, and then translate the datasets into the first-order logic representation above, along with their ontologies and the alignment axioms relating the ontologies. Then apply a first-order reasoner to determine whether or not the combined axioms are consistent. The merge of two datasets $\Phi_M$ is the union of the formulas for each dataset combined with the formulas derived from the RCC-5 articulations between the dataset ontologies

$$\Phi_M = \Phi_{DS_1} \cup \Phi_{DS_2} \cup \Phi_{A_1} \cup \cdots \cup \Phi_{A_n}$$

where $n$ ranges over the context attributes in the datasets.

**Example (Merge-Compatible).** Consider Fig. 7.1 without absence closure, and ontology alignment set $\mathcal{A} = \{\{A \equiv E; B \equiv F; C \equiv G\}, \{J \equiv M; K \equiv M; L \equiv M\}, \{R \equiv S\}\}$. In this simple example, merging the two datasets is straightforward, where the single merge result shown in Table 7.1(c) contains no BRU or DRU and represents all the observed data. Typically, however, merging two datasets does not result in a combined dataset that is free of uncertainty, due to non-trivial ontologies and articulation constraints. The following section describes an approach for merging datasets when the merge cannot be satisfied by

a single dataset, and instead must be represented as a set of possible merges.

## 7.4 Merging Data Sets

Merging two datasets results in a set of possible merges, each representing an unambiguous dataset that respects the observations in the source datasets. Before carrying out the merge, determine the input datasets' merge compatibility. If the sets are merge compatible, perform one of two types of merge. *Basic relation merges* (BRM) are those in which all the relations between concepts in the two datasets are drawn from the basic set relations. *Disjunctive relation merges* (DRM) are those that involve at least one disjunctive relation (e.g., $A\{\equiv\subsetneq\}B$). This section proceeds by first describing how to check for merge compatibility, followed by a description of a naive algorithm for merging datasets. The section ends with two BRM algorithms and a description of how to perform a DRM.

### 7.4.1 Merge Compatibility and Absence Closure

For two datasets to be merge compatible, they must follow the given schema, their ontologies must be consistent, the data must be consistent with the ontologies, the alignments between their ontologies must be consistent, and finally, the union of the logic axioms for each dataset, their ontologies, and the ontology alignments must be consistent. These steps are outlined in Algorithm 1.

Consistency in the last step may be violated by contradictions introduced by explicit absence statements, as well as axioms introduced in absence closure. For example, in Fig. 7.1, if $D \equiv X$ where $X$ is some known, but unreported rodent in dataset 2's taxonomy, absence closure leads to a direct contradiction; dataset 2 would state explicitly that $X$ is absent, conflicting with the observed $D$ in dataset 1. Algorithm 2 provides a straightforward way of calculating these absence axioms. This algorithm first determines all possible cases in which presence might be observed within the given attribute contexts, and then rules out those cases that are implied by known observations, and also those that imply known

**Algorithm 1: Merge Compatible**
*Input*: Two datasets and a set of articulations between the ontologies
*Output*: `true` if the datasets are merge compatible, `false` otherwise

1. Determine consistency.
   (a) For each dataset
      i. Calculate $\Phi_O$ for each ontology and check its consistency.
      ii. Calculate $\Phi_{DS}$ for the dataset and check its consistency.
2. If each dataset is consistent, check the alignment $\Phi_{O_1} \cup \Phi_{O_2} \cup \Phi_{A_{12}}$ between each pair of dataset ontologies for consistency.
3. If each alignment is consistent, check the full merge $\Phi_M$ for consistency, applying absence closure if required.

**Algorithm 2: Calculate Absence Closure Axioms**
*Input:* A dataset
*Output:* A set of logic axioms representing absence axioms

1. Create logic absence axioms $A = \{a_1, \cdots a_n\}$ for each possible combination of context attribute values $B \times S \times T$
2. For each row in the dataset $r_i$, for each created absence axiom $a_i$:
   (a) if $r_i \rightarrow a_i$ remove $a_i$ from $A$
   (b) if $a_i \rightarrow r_i$ remove $a_i$ from $A$
3. Return $A$

observations.

### 7.4.2 The Naive Basic Relation Merge Algorithm

The most straightforward way to calculate the possible worlds is to create an initial world set (IWS) as described in Section 7.6.2, encode each world in logic, and test whether or not it is consistent with the formulas in $\Phi_M$. This method, however, is both intractable and inefficient. A somewhat more efficient approach is to initially rule out impossible conditions in the IWS. For example, if an articulation holds that $A \subsetneq B$, any world in which the combined concept $A\bar{B}$ is either present or absent would be inconsistent with the articulation. Removing conditions containing such concepts reduces the size of the IWS and,

as will be shown in Section 7.5, can generate possible worlds for small datasets. Table 7.3 lists the monadic logic formulas generated to test the possible world in Fig. 7.2(d) in which instances of taxa that are both $A$ and $B$ are present, as well as instances of taxa that are $B$ but not $A$. The complexity of naive BRM algorithm comes primarily from the need to perform many (up to $2^n$) monadic logic proofs, each of which is NEXPTIME [BGW93].

| Axioms: | | Conjecture: |
|---|---|---|
| $\forall x : A(x) \rightarrow B(x).$ | $\exists x : AB(x).$ | $(\forall x : (A(x) \rightarrow B(x))) \wedge$ |
| $\forall x : B(x) \leftrightarrow (AB(x) \vee \bar{A}B(x)).$ | $\exists x : \bar{A}B(x).$ | $(\exists x : A(x)) \wedge (\exists x : B(x)).$ |
| $\forall x : A(x) \leftrightarrow AB(x).$ | $\neg\exists x : \bar{A}\bar{B}(x).$ | |
| $\forall x : A(x) \vee B(x).$ | $\forall x : \bar{A}B(x) \rightarrow (\neg A(x) \wedge B(x)).$ | |
| $\forall x : AB(x) \rightarrow (A(x) \wedge B(x)).$ | $\forall x : \bar{A}\bar{B}(x) \rightarrow (\neg A(x) \wedge \neg B(x)).$ | |

Table 7.3: Monadic logic rules demonstrating the possibility of the dataset in Fig. 7.2(d).

### 7.4.3 General Basic Relation Merge (BRM-G)

The general basic relation merge (BRM-G) presented in Algorithm 3 applies when the datasets to be merged have no DRM, but may have BRM. The key steps to the BRM-G algorithm are calculating the columns of the PWS $H$, and the propositional formula $\Phi$, the models of which represent the possible worlds in the PWS. The compress function in step 2(c)ii takes two combined concepts, both of length $n$. If the two combined concepts agree on $n-1$ concepts, the result is the concepts they agree on, plus the concepts they disagree on. For example, $compress(A\bar{B}\bar{C}, \bar{B}\bar{C}D)$ results in A$\bar{B}\bar{C}$D. The compress function also makes sure to not create any impossible combined concepts, such as ones that contain a term and its negation (e.g., $A\bar{A}$).

**Example (Only one context domain).** Consider a simplified version of Fig. 7.1 with only the biological attribute context, the observation data context, and the following alignment between the biological ontologies of the datasets: $\mathcal{A} = \{A \equiv E; B \equiv F; C \oplus G; D \subsetneq G\}$. A straightforward union of the biological concepts in this situation shown in Table 7.4(a) contains several problems. First, listing both $A$ and $E$ is redundant, as $A \equiv E$. More importantly, $D$ and $G$ have a $\subsetneq$ relation between them, so the result in Table 7.4(a) still

**Algorithm 3: General Basic Relation Merge (BRM-G)**
*Input*: A naively merged dataset.
*Output*: A possible world set representing each possible merge.

1. Create a new concept $c_1c_2\cdots c_n$ for those concepts that are equivalent according to the articulations. Replace all concepts contributing to the new concept with the new concept in $\Phi_M$. Remove redundant formulas.
2. For each attribute $A \in \{B, S, T\}$:
   (a) Create an empty set $P_A$.
   (b) For each pair of rows $(r_i, r_j)$ in the dataset
      i. Let $c_i = A(r_i)$ and $c_j = A(r_j)$
      ii. If $c_i \subsetneq c_j$, add $c_ic_j$ and $\bar{c_i}c_j$ to $P_A$.
      iii. if $A(c_i) \oplus A(c_j)$, add $c_ic_j$, $c_i\bar{c_j}$, $\bar{c_i}c_j$ to $P_A$.
      iv. if $A(c_i)\,!\,A(c_j)$, add $c_i\bar{c_j}$ and $\bar{c_i}c_j$ to $P_A$.
   (c) Repeat $|A| - 2$ times, where $|A|$ is the number of concepts of attribute A in the datasets.
      i. Create empty set $E_A$
      ii. For each pair of concepts $c_i, c_j, i \neq j \in P_A$, add $compress(c_i, c_j)$ to $E_A$
      iii. set $P_A = E_A$
3. For each dataset row $r$, for each attribute $A \in \{B, S, T\}$, for each term $p \in P_A$, if $A(r)$ appears positively in $p$ then add $p$ to $V_{A(r)}$.
4. Create a propositional logic statement that will generate the possible worlds: Create empty sets $A$ and $H$. For each observation $r$ in each dataset:
   (a) For each attribute $A \in \{B, S, T\} : D_A = \bigvee V_{A(r)}$
   (b) $C = \bigvee D_B \times D_S \times D_T$
   (c) If $O(r) = \texttt{P}$, add $C$ to $A$
   (d) If $O(r) = \texttt{N}$ add the negation of $C$ to $A$
   (e) Add $C$ to $H$
5. Conjoin the elements in set $A$ – this will be a propositional logic statement – the possible worlds are the models of this statement.
   (a) $H$ contains the conditions in the header of the table
   (b) Create the rows: For each model, add a new row to the table where for each condition in $H$, if the condition holds in the model, put 1 in the appropriate column, and add 0 otherwise.

$$H = \{AE, BF, C\bar{G}, CG, \bar{C}\bar{D}G, DG\}$$
$$\Phi = AE \wedge \neg BF \wedge (C\bar{G} \vee CG) \wedge (\bar{C}\bar{D}G \vee CG \vee DG) \wedge \neg DG$$

| Taxon | O |
|---|---|
| $A$ | P |
| $B$ | N |
| $CG$ | P |
| $D$ | N |
| $E$ | P |
| $F$ | N |

(a)

| **World** | **AE** | **BF** | **C$\bar{\text{G}}$** | **CG** | **$\bar{\text{C}}\bar{\text{D}}$G** | **DG** |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 |
| 5 | 1 | 0 | 0 | 1 | 0 | 0 |

(b)

| Taxon | O |
|---|---|
| $AE$ | P |
| $BF$ | N |
| $C\bar{G}$ | P |
| $CG$ | N |
| $\bar{C}\bar{D}G$ | P |
| $DG$ | N |

(c)

| Taxon | O |
|---|---|
| $AE$ | P |
| $BF$ | N |
| $C\bar{G}$ | N |
| $CG$ | P |
| $\bar{C}\bar{D}G$ | N |
| $DG$ | N |

(d)

Table 7.4: Possible worlds for Fig. 7.1 with just its biological attribute context and its data context. (a) shows a merge representing the (ambiguous) straightforward union of the datasets, (b) shows the PWS of unambiguous worlds. Tables (c) and (d) represent unambiguous merged datasets derived from the PWS.

contains BRU. Finally, although $C$ and $G$ are both named pack rat, they are not equivalent terms as represented in Table 7.4(a).

Running the BRM-G against this example results in the $H$ and $\Phi$ shown at the top of Table 7.4.[5] The PWS that results from these formulas is shown in Table 7.4(b). The enumeration of all possible worlds shown in Table 7.4(b) indicates that the combined concept $AE$ is present in all possible worlds (certainly present), while $BF$ and $DG$ are absent in all possible worlds (certainly absent). The situation is more complicated for concepts $C$ and $G$. Table 7.4(c) and (d) give two of the possible merged datasets showing different possible configurations of $C$ and $G$ .

[5]To save space and improve legibility, the complete combined concepts are not given in the table. For each abbreviated concept in Table 7.4, the full combined concept can be determined by adding the negated form of all the concepts in the datasets not mentioned in the combined concept. For example, the abbreviated combined concept $AE$ in Table 7.4 stands for $A\bar{B}\bar{C}\bar{D}E\bar{F}\bar{G}$.

**Example (Two context domains).** Consider the taxonomic and spatial dimensions of the running example with the alignment $\mathcal{A} = \{\{A \equiv E; B \equiv F; C \oplus G; D \subsetneq G\}, \{J \subsetneq M; K \subsetneq M; L \subsetneq M\}\}$. Below are the columns of the PWS (each given a number), followed by the propositional formula describing the possible worlds.

$$\begin{aligned} H \quad = \quad & \{1 : AEJM, 2 : BFKM, 3 : CGJM, 4 : C\bar{G}JM, 5 : DGCM, 6 : AEKM, 7 : AELM, 8 : AE\bar{J}\bar{K}\bar{L}M, \\ & 9 : BFJM, 10 : BFKM, 11 : BFLM, 12 : BF\bar{J}\bar{K}\bar{L}M, 13 : CGKM, 14 : CGLM, 15 : CG\bar{J}\bar{K}\bar{L}M \\ & 16 : \bar{C}\bar{D}GJM, 17 : \bar{C}\bar{D}GKM, 18 : \bar{C}\bar{D}GLM, 19 : \bar{C}\bar{D}G\bar{J}\bar{K}\bar{L}M, 20 : DGJM, 21 : DGKM, 22 : DG\bar{J}\bar{K}\bar{L}M\} \end{aligned}$$

$$\begin{aligned} \Phi \quad = \quad & 1 \wedge \neg 2 \wedge (3 \vee 4) \wedge \neg 5 \wedge (1 \vee 6 \vee 7 \vee 8) \wedge \neg (9 \vee 10 \vee 11 \vee 12) \wedge \\ & (3 \vee 13 \vee 14 \vee 15 \vee 16 \vee 17 \vee 18 \vee 19 \vee 20 \vee 21 \vee 5 \vee 22) \end{aligned}$$

$\Phi$ has 24576 ($< 2^{15}$) models, each of which is a possible merged dataset. This may seem like a large number, but it is considerably smaller than the number of possible worlds in the initial world set ($2^{2^{(7+5)}} = 2^{4096}$). The BRM-G is considerably more efficient than the naive algorithm because it involves a single NP-complete SAT proof, rather than up to $2^n$ NEXPTIME monadic logic proofs. The algorithm itself, however is $O(2^n)$ due to the need to run the compress function multiple times. Each time compress is run, the size of $P$ changes, and in the worst case, when all the concepts overlap, $|P| = 2^n$ the final time compress is run.

### 7.4.4 The Basic Relation Merge for Unambiguous Data Sets (BRM-U)

The BRM-G algorithm works when source datasets contain BRU. The BRM-U algorithm presented here is far more efficient, but only works when the source datasets have no BRU (or DRU). The only difference between the BRM-G and BRM-U algorithms is in how the compress function creates combined concepts. In the BRM-G algorithm, compress is run $n-2$ times where $n$ is the number of distinct concepts in the source datasets and the input can be as large as $2^n$ combined concepts. In the BRM-U algorithm, on the other hand,

compress is only run once on $\binom{n}{2}$ combined concepts. This is possible because when a dataset has no BRU, and the equivalent concepts have been combined into a single concept (step 1 in Algorithm 3), any combined concept can contain at most one pair of non-negated concepts (one concept from each dataset). After a single run of compress, each combined concept in $P_A$ will be three concepts long, and all the feasible pairs of non-negated concepts will have been found. After this single run of compress, each combined concept is then padded with the negated version of all the $n-3$ concepts that are not yet in that combined concept. The resulting compress algorithm is $O(n^2)$ as it involves a single pass through $\binom{n}{2}$ combined concepts determined in step 2b of Algorithm 3. The entire BRM-U algorithm is $O(n^2)$ except for the single SAT proof at the end, which is NP-complete.

### 7.4.5 Merging under Disjunctive Relation Uncertainty

The algorithm described here applies to merges involving both BRU and DRU. The strategy is to divide alignments containing disjunctions into several alignments containing no disjunctions, determine the PWS for each BRM situation, and combine the results. Dividing disjunction containing alignments into several basic alignments is an expensive process. Consider, e.g., the taxonomy alignment in Fig. 7.3, which contains two disjunctive relationships {A {≡,⊊} E ; B *e.g.* F} and represents "isa" relations as ⊊. To decompose this disjunction-containing alignment into alignments containing only basic relations, one might try simply multiplying out the disjunctive relationships, creating four possible alignments. If, however, the following additional constraints hold in the alignment: $X \equiv A \vee B \vee C \vee D$; $Y \equiv E \vee F \vee G$, and sibling concepts are disjoint, two of the four possible alignments ($\{A \subsetneq E; B \equiv F\}$ and $\{A \equiv E; B \supsetneq F\}$) are ruled out.

With this in mind, the disjunction containing alignment above can be divided into two consistent alignments containing only basic relations: one equivalent to the one described in Section 7.4.3, and the other following the alignment: $\mathcal{A} = \{A \subsetneq E; B \supsetneq F; C \oplus G; D \subsetneq G\}$. Table 7.5(a) shows the complete PWS for the disjunction containing alignment. The column to the side of the table records which alignment applies to the given row: alignment 1 is

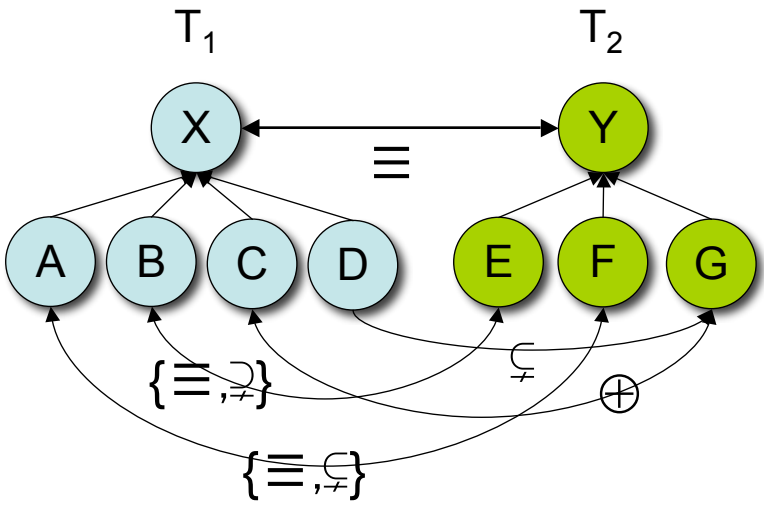



Figure 7.3: When sibling concepts are disjoint and parents contain no instances not found in their children, this disjunctive relation containing alignment has two basic relation interpretations.

where $A \equiv E$ and $B \equiv F$, and alignment 2 is where $A \subsetneq E$ and $B \supsetneq F$. This additional information may be considered provenance; the actual metadata for the merged datasets is still the merged ontologies of the original datasets. The $\perp$ seen in worlds 1 through 5 indicates that the combined concept does not exist in that world.

Table 7.5 contains some subtly different merges. For example, in merge 15 (shown in Table 7.5(c)), no instances of $\bar{A}E$ were seen, while in merge 5 (shown in Table 7.5(b)) there is no such thing as an instance of $\bar{A}E$.

## 7.5 Evaluation

This section evaluates the efficiency of the basic relation merge, which is the core of our dataset merging methodology. The naive, BRM-G, and BRM-U dataset merge algorithms are compared here using two types of datasets: those containing no BRU or DRU (the *unambiguous inputs* condition), and those that contained BRU (the *ambiguous inputs* condition).

Pairs of datasets and the alignments between their concepts were generated randomly.

| World | AE | BF | $\bar{A}E$ | $B\bar{F}$ | $C\bar{G}$ | CG | $\bar{C}\bar{D}G$ | DG | P |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | ⊥ | ⊥ | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | ⊥ | ⊥ | 1 | 0 | 1 | 0 | 1 |
| 3 | 1 | 0 | ⊥ | ⊥ | 1 | 1 | 0 | 0 | 1 |
| 4 | 1 | 0 | ⊥ | ⊥ | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 0 | ⊥ | ⊥ | 0 | 1 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 2 |
| 7 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 |
| 8 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 |
| 9 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 2 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2 |
| 11 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 |
| 12 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 13 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 2 |
| 14 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 |
| 15 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 |

(a)

| Taxon | O |
|---|---|
| $AE$ | P |
| $BF$ | N |
| $C\bar{G}$ | N |
| $CG$ | P |
| $\bar{C}G$ | N |
| $DG$ | N |

(b)

| Taxon | O |
|---|---|
| $AE$ | P |
| $BF$ | N |
| $\bar{A}E$ | N |
| $B\bar{F}$ | N |
| $C\bar{G}$ | N |
| $CG$ | P |
| $\bar{C}G$ | N |
| $DG$ | N |

(c)

Table 7.5: PWS for Section 7.4.5(a) and two datasets derived from the PWS: world 5 in (b) and world 15 in (c).

Each dataset had only one context attribute, and each observation of the datasets was recorded as present. Generating interesting large consistent alignments between dataset concepts proved challenging. While it is simple to generate large alignments in which all the relations are of one type (e.g., all equivalent, all overlapping, all disjoint), generating consistent alignments that mix relations is computationally expensive. To address this issue, alignments of up to 9 concepts in which the non-disjoint relationships were either all-$\oplus$, all-$\subsetneq$, or had mixed relations, including $\equiv$, $\oplus$, $\subsetneq$, and $\supsetneq$ were generated. The same patterns of results held in the all-$\oplus$, all-$\subsetneq$, and mixed conditions, so for datasets of fewer than 10 concepts, the average results of these three types of datasets are reported. Results

| **Data Items** | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|
| **naive** | 8.83 | 23.57 | 32.49 | > 60 | > 60 | > 60 | > 60 |
| **BRM-G** | 0.03 | 0.04 | 0.08 | 0.34 | 1.86 | 16.88 | 23.91 |

(a) Ambiguous Inputs

| **Data Items** | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|
| **naive** | 1.73 | 6.46 | 18.09 | > 60 | > 60 | > 60 | > 60 |
| **BRM-G** | 0.03 | 0.04 | 0.05 | 0.11 | 0.37 | 2.06 | 7.24 |
| **BRM-U** | 0.03 | 0.04 | 0.04 | 0.05 | 0.05 | 0.06 | 0.06 |

(b) Unambiguous Inputs

| Data Items | 25 | 50 | 75 | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|---|---|---|
| **BRM-U** | 0.19 | 0.37 | 0.86 | 2.32 | 24.54 | 121.25 | 359.24 | 824.92 |

(c) BRM-U with larger unambiguous input datasets

| **Data Items** | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|
| **Ambiguous Inputs** | 22 | 12 | 26 | 81 | 266 | 173 | 180 |
| **Unambiguous Inputs** | 2 | 3 | 7 | 86 | 58 | 165 | 224 |

(d) Worlds generated by mixed relation datasets

Table 7.6: Average run times in seconds for the naive algorithm and two versions of the BRM algorithm using datasets of between 3 and 9 concepts in two conditions: (a) where the dataset contains basic relation uncertainty, and (b) where the input datasets do not contain basic relation uncertainty. Run times in seconds for larger datasets using the BRM-U algorithm are shown in (c). The average number of worlds generated by datasets with mixed relations is shown in (d).

of 10 or more concepts are the average of the all-$\oplus$ and all-$\subsetneq$ conditions. Each condition was run three times, and only marginal variance was found between runs.

The naive algorithm runs employed the first-order reasoner iProver 0.7 [Kor08] to test whether a given world qualifies as a possible merge. Comparisons between iProver and several other available first-order reasoners showed iProver to be the fastest to solve our class of problem. The BRM algorithm tests employed the c2d [Dar04] reasoner to check the satisfiability of the propositional statement that determines the possible merges, and to generate and count models of the statement. c2d has the advantage of providing polynomial-time model counting.

As may be seen in Table 7.6, the naive implementation performs poorly, taking over a minute to generate possible worlds for datasets with more than 6 concepts. In the ambiguous input condition, the BRM-G algorithm performs considerably better. However, the time

to generate the possible worlds still grows exponentially with the size of the input. The unambiguous inputs condition shows the same pattern for the naive and general BRM algorithms. However, the BRM-U algorithm performs comparatively well, providing both a feasible and efficient method for generating the possible dataset merges. Table 7.6(c) shows how the BRM-U algorithm scales to up to 500 concepts. The presence datasets found in MetaCat [BJBM01] have listed fewer than 300 concepts, and the largest pair of articulated biological taxonomies we have seen to date [Pee05] has comprised 360 concepts, so the algorithm scales well to the currently available real-world data. Table 7.6(d) gives the average number of worlds generated by the datasets with mixed relations.

## 7.6 Towards a Best-Effort Merge of Taxonomically Organized Data

As discussed, it may not be possible to derive a single merged dataset that adheres to the specificity constraint, and so it may be necessary to provide multiple possible data merges that adhere to the constraint. If, however, a user desires only one merged dataset, we will need to provide a *best-effort merge* which is as specific as possible.

This section investigates the applicability of the *data exchange* setting to the problem of determining a single best-effort merge.

### 7.6.1 Introduction

In *data exchange*, a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$ are given, together with source-to-target dependencies $\Sigma_{st}$ and target schema constraints $\Sigma_t$. Given an input instance $I$ of $\mathbf{S}$, a *solution* to the data exchange problem is a target instance $J$ of $\mathbf{T}$, such that $\langle I, J\rangle$ satisfies $\Sigma_{st}$ and $J$ satisfies $\Sigma_t$. In general, there are multiple solutions, so the *certain* answers, i.e., contained in all possible solutions $J$, are usually reported. Data exchange has been well studied in recent years [FKP03, FKPT07, AL08] and tractable algorithms for many common scenarios have been developed.

Although there are some potential mismatches between our setting and that of traditional data exchange, we can still use the machinery of data exchange to help solve the problem of finding the best-effort merge. The following section sketches out this process.

### 7.6.2 Approach

Mapping our merge scenario into the data exchange schema involves several pieces. These include (i) describing how a dataset is translated into a source instance, (ii) determining the source schema, (iii) determining the target schema constraints $\Sigma_t$, (iv) deriving the source-to-target dependencies $\Sigma_{st}$, and (v), describing how the target instance calculated using data exchange is translated into a dataset that satisfies the specificity constraint.

#### A Simple First Attempt

A first, straightforward translation is to take all concepts $\mathsf{C}$ in the input taxonomies $\mathsf{T}_1, \mathsf{T}_2$ and view them as unary relations $\mathtt{C}(\mathtt{x})$ of the source schema $\mathbf{S}$, and consider concepts $\mathsf{C}'$ of the merged taxonomy $\mathsf{T}_3$ as relations $\mathtt{C}'(\mathtt{x})$ of the target schema $\mathbf{T}$. For presence/absence datasets in their most basic form, we can only state that a concept $\mathsf{C}$ is present or absent. We model this via facts of the form $\mathtt{C}(\mathtt{P})$ and $\mathtt{C}(\mathtt{N})$, representing that some instance of concept $\mathsf{C}$ was (or wasn't) observed.

**Translating Taxonomies to Schemas** The next question is how to represent the taxonomy constraints. Using RCC-5, for any two non-empty sets $\mathsf{A}$, $\mathsf{B}$, exactly one of the following relations must hold: $\mathsf{A} \equiv \mathsf{B}$, $\mathsf{A} \subsetneq \mathsf{B}$, $\mathsf{A} \supsetneq \mathsf{B}$, $\mathsf{A} \oplus \mathsf{B}$, or $\mathsf{A} \,!\, \mathsf{B}$. Proper part ($\subsetneq$) and overlaps ($\oplus$) were described in the introduction. Identity ($\mathsf{A} \equiv \mathsf{B}$) is defined as $\forall x : \mathsf{A}(x) \leftrightarrow \mathsf{B}(x)$. The last relation ($\mathsf{A} \,!\, \mathsf{B}$) refers to disjointness and can be represented as $\neg\exists x : \mathsf{A}(x) \wedge \mathsf{B}(x)$. Identity and proper part constraints can be approximated by the following constraints:

$$\mathsf{A} \equiv \mathsf{B} : \mathtt{A}(\mathtt{P}) \leftrightarrow \mathtt{B}(\mathtt{P}),\ \mathtt{A}(\mathtt{N}) \leftrightarrow \mathtt{B}(\mathtt{N})$$

$$\mathsf{A} \subsetneq \mathsf{B} : \mathtt{A}(\mathtt{P}) \rightarrow \mathtt{B}(\mathtt{P}),\ \mathtt{B}(\mathtt{N}) \rightarrow \mathtt{A}(\mathtt{N})$$

These formulas describe integrity constraints on a schema **S**. The first states that if $\mathsf{A} \equiv \mathsf{B}$ in the taxonomy, then $\mathtt{A}(\mathtt{P}) \in I$ iff $\mathtt{B}(\mathtt{P}) \in I$, and $\mathtt{A}(\mathtt{N}) \in I$ iff $\mathtt{B}(\mathtt{N}) \in I$. The second constraint states that if $\mathsf{A} \subsetneq \mathsf{B}$ in the taxonomy, then if taxon A has been reported as present in the dataset, then taxon B must also be present. In addition, if B has been reported as not present in the dataset, then A must also be reported as not present. Note that this is essentially an encoding in propositional logic, and that some information is lost in the process. For example, the disjointness relation does not constrain the schema in our case; even though A and B are disjoint, examples of each may be observed. Given this simple translation of concepts into relations, the partially overlaps relation does not translate into an integrity constraint.

**Translating Datasets into Instances** Datasets rarely contain information about all the nodes in the accompanying taxonomy. However, given a source schema as translated above, a dataset may be used to populate much of the schema. For example, if $\mathsf{A} \subsetneq \mathsf{B}$ and $\mathsf{B} \subsetneq \mathsf{C}$ and the dataset contains (A, P), we can assert the presence of B and C. Similarly, if the dataset contains (C,N), we assert A(N) and B(N). This information can be used to determine inconsistent datasets. If a relation contains both P and N tuples, it is inconsistent.

**Calculating $\Sigma_{st}$** In this simple scenario, $\Sigma_{st}$ is given by the constraints above.

**Translating Instances Back into Datasets** Once the data exchange setting is constructed, we can calculate a merged dataset by taking the source datasets, representing them as instances of the source schema, and following the dependencies to construct a target instance. To finish the process, that target instance should be translated back into a dataset.

As discussed in the introduction, we want to avoid ambiguity in our datasets. Unfortunately, the target instance as it stands will almost always be ambiguous. Even if there is only one dataset, with one observation "A is present", if $\mathsf{A} \subsetneq \mathsf{B}$, then the target instance calculated via data exchange will have A(P) and B(P). The B(P) is ambiguous because, once

it is taken out of the context of the data exchange setting, it is unclear whether that $\mathtt{B}(\mathtt{P})$ is due to the $\mathtt{A}(\mathtt{P})$ or if there is some other $\mathsf{B}$ present which is not also an $\mathsf{A}$.

One potential solution would be to only include the leaves of the merged taxonomy when translating from a target instance to a target dataset. However, in many cases this approach would lead to incorrect results. For example, consider two taxonomies of one concept each, $\mathsf{A}$ in $\mathsf{T}_1$ and $\mathsf{B}$ in $\mathsf{T}_2$. Dataset $D_1$, registered to $\mathsf{T}_1$, reports the presence of $\mathsf{A}$, dataset $D_2$, registered to $\mathsf{T}_2$, reports the presence of $\mathsf{B}$. The target instance contains $\mathtt{A}(\mathtt{P})$ and $\mathtt{B}(\mathtt{P})$, but converting these into a target dataset that only contains $\mathsf{A}$ would be incorrect. In particular, it may be the case that something that was a $\mathsf{B}$ but not an $\mathsf{A}$ was present.

**Problems with the First Attempt** For data exchange to be useful, the target instance we generate either must adhere to the specificity constraint, or it must at least provide us with enough information to derive a dataset that adheres to the constraint.

There are many problems with the naive translation above, e.g., it is hard or impossible to enforce the specificity constraint when translating the target instance back into a dataset. The key difficulty in enforcing the constraint is that statements like "something that is a $\mathsf{B}$ but not also an $\mathsf{A}$" cannot be represented if we restrict ourselves to discussing concepts from the taxonomies. Instead, to unambiguously describe something that is a $\mathsf{B}$ but not an $\mathsf{A}$ requires combined concepts.

### Combined Concepts and Disjunctive Constraints

Allowing the use of combined concepts in the target schema permits us to use the specificity added by the taxonomy alignments. For example, if $\mathsf{A}$ overlaps $\mathsf{B}$, the combined concepts are $\mathsf{AB}$, $\mathsf{A}\bar{\mathsf{B}}$, $\bar{\mathsf{A}}\mathsf{B}$, and the source to the target dependencies are:

$$\mathtt{A}(\mathtt{P}) \rightarrow \mathtt{A}\bar{\mathtt{B}}(\mathtt{P}) \vee \mathtt{AB}(\mathtt{P})$$

$$\mathtt{A}(\mathtt{N}) \rightarrow \mathtt{A}\bar{\mathtt{B}}(\mathtt{N}) \wedge \mathtt{AB}(\mathtt{N})$$

$$\mathtt{B}(\mathtt{P}) \rightarrow \bar{\mathtt{A}}\mathtt{B}(\mathtt{P}) \vee \mathtt{AB}(\mathtt{P})$$

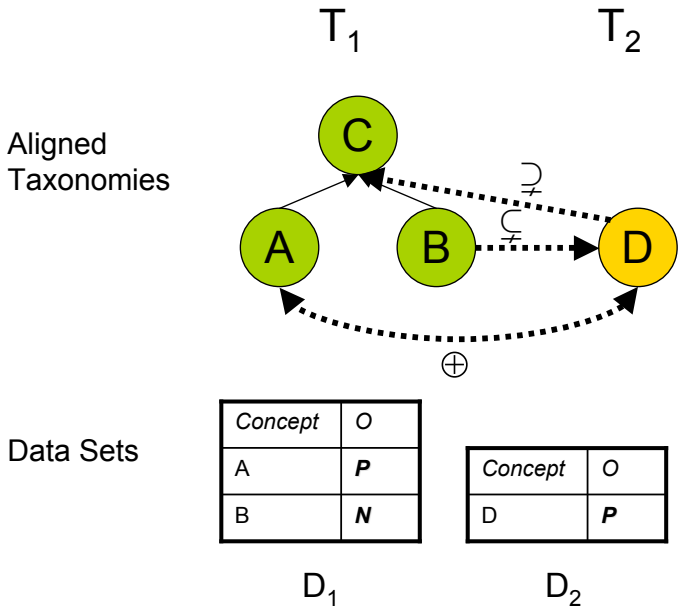



Figure 7.4: Aligned taxonomies $T_1, T_2$ with datasets to be merged.

$$\mathtt{B(N)} \rightarrow \bar{\mathtt{A}}\mathtt{B(N)} \wedge \mathtt{AB(N)}$$

These disjunctive dependencies result in a target instance that contains uncertainty [FKPT07]. We can query this target instance, e.g., to determine if each combined concept's presence or absence value is certainly known or not. Alternatively, we can materialize all possible instances. Tables 7.7(a) and (b), for example, show two of the five possible merges[6] resulting from Figure 7.4. Each of the concepts described in Tables 7.7(a) and (b) is a most specific applicable concept: it specifies whether or not it is subsumed by each of the original concepts. For example, the concept $\mathsf{A\bar{B}C\bar{D}}$ is subsumed by concepts A and C, but not subsumed by concept B or D.

### Constructing the Most Specific Single World

The target instance follows a schema derived from the merged taxonomies. If the target instance contains uncertainty, (e.g., instances of the combined concept $\mathsf{A\bar{B}C\bar{D}}$ may or may

[6]There are five possible merges under the assumptions that concepts A and B are disjoint and that there are no instances in concept C that are not in either A or B.

| *Concept* | *O* |
|---|---|
| $A\bar{B}C\bar{D}$ | N |
| $A\bar{B}CD$ | P |
| $\bar{A}BCD$ | N |
| $\bar{A}\bar{B}CD$ | N |

(a)

| *Concept* | *O* |
|---|---|
| $A\bar{B}C\bar{D}$ | P |
| $A\bar{B}CD$ | N |
| $\bar{A}BCD$ | N |
| $\bar{A}\bar{B}CD$ | P |

(b)

| *Concept* | *O* |
|---|---|
| C | P |

(c)

Table 7.7: Two possible merges (a), (b) of the datasets in Figure 7.4. A single best-effort merge is shown in (c).

not be present) we do not have a single most specific world. One way to get a most specific single world is to alter the merged taxonomy by removing concepts that contribute to the uncertainty. If uncertainty arises when internal (non-leaf) concepts are included in the target dataset, we can address this issue by removing all the leaves under the problematic internal concepts. This in effect makes the merged taxonomy less specific, but makes the dataset unambiguous in the context of the less specific taxonomy.

Table 7.7(c) provides an example of a single best-effort merge for the scenario in Figure 7.4.

### 7.6.3 Some Challenges for the Best-Effort Merge

This section provides a first, very rough sketch of how to leverage data exchange to merge datasets that draw their concepts from taxonomies. Fleshing the process out more completely and giving it a formal foundation is the first priority in future work on the problem. Assuming that the process described here can be used to create a best-effort merge, further issues arise:

**Disjunctive Relations** In this section we have restricted our articulation relations to the basic five set relations. RCC-5 also encodes disjunctive relations, e.g., $(\mathsf{A} \subsetneq \mathsf{B}) \vee (\mathsf{A} \equiv \mathsf{B})$ is the standard definition of "isa". There are 32 disjunctions over the basic five relations.

It is unclear how to extend the approach as given to a scenario where the taxonomies, or the articulations between the taxonomies, contain such disjunctions.

**Restricting the Target Instance to Original Concept Terms** An effect of our move to combined concepts is that the resulting merged dataset will most likely use a vocabulary different from the vocabulary used in the original datasets. This calls for a second translation from the calculated dataset to one using only terms from the original taxonomies. Although this conversion should be fairly straight-forward, an algorithm for this conversion has not yet been developed.

**Translation from the Target Instance to a Dataset** It remains to be seen how to derive from the target instance a merged dataset that adheres to the specificity constraint, and how to encode the constraint as dependencies on the target instance.

## 7.7 Related Work and Conclusion

This chapter has described a framework and algorithms for merging data sets when the domains of attributes overlap and contain uncertainty. The chapter has shown that no single merge, except in trivial cases, can satisfy all the requirements of a dataset merge, and multiple merges must be represented. The chapter has given a possible worlds semantics for such datasets, and algorithms for constructing these possible worlds when ambiguity arises during the merge. The chapter has also presented an efficient algorithm for performing the merge when ambiguity is due to articulations (i.e., source datasets do not contain ambiguity).

The three areas most similar to the current work are traditional data integration, data fusion, and ontology merging. In traditional data integration [Len02], two or more databases with different schemas are combined through the definition of a common (i.e., global) schema. The current work, on the other hand, focuses on merging datasets when the schemas of datasets are the same, but the domains of the schema attributes may be different.

Another difference is that in traditional data integration, the data themselves are generally not considered; integration happens at the schema level. In the current work, however, the alignments between the domains of the dataset attributes impact the interpretation of the data itself. Data fusion [VMP05] tasks typically involve integrating multiple types of information about the same objects. The data fusion setting differs from the current one in that we are merging datasets that contain the same type of data: presence data, in this case. Furthermore, our observations are about sets of objects rather than individuals. Ontology merging [NM00, MFRW00a, SM01b], like traditional data integration tasks, focuses on the schema level rather than the instance level. The work in Chapter 6, which describes how to merge taxonomies that have been aligned with RCC-5 relations, is more similar to ontology merging. As we have seen here, merging taxonomies is just the first step in merging taxonomically organized datasets.

This work can be expanded in several directions. First, although we use RCC-5 to describe relations between attribute domains, there are other algebras that may be more suited to specific domains. For example, RCC-8 may be a better language to describe relations between spatial regions. Allen's interval calculus is more suited for the temporal dimension. The types of languages used constrain the questions that may be asked. In this work, we are satisfied to ask questions that are suitable for RCC-5 articulated domains. In the future, other languages should be applied. Second, in the current work, domains are independent. However, in general this may not be the case. For example, one taxonomic alignment may apply in one spatial region, while a second taxonomic alignment may apply in a different region. Extending the algorithms to deal with this extra complexity is not straightforward. Third, we have only considered presence data here. As we have seen, merging datasets with even this limited type of data is complicated. However, datasets typically contain data other than simple presence data, so this work should be extended to include other types of measurements. Fourth, the work on constructing a single best-effort merge should be continued. Finally, the work must be evaluated by testing its utility for the people who currently spend their time integrating datasets by hand. This evaluation

will no doubt generate interesting new avenues of study.

# Chapter 8

# Implementations

## 8.1 Overview and Objectives

The operations and representations described throughout this thesis may be implemented in many ways. Described below are two implementations. The first, a traditional object-oriented implementation written in Python, is geared toward the interests of a working metadata curator. It integrates reasoners, databases, and visualization tools to assist a metadata curator in analyzing and creating taxonomies and articulations between them. The second implementation is more geared toward researchers who might want to experiment with different reasoners, visualizations and reports. This system breaks the first system into a number of components that may be tied together within a workflow system. The proposed implementation uses the Kepler workflow system to create a framework to make it easy for researchers to experiment with alternative reasoners and algorithms. Before launching into the implementations, however, there are a fair number of details common to both. These are described first.

## 8.2 Features Common to All Implementations

At a high level, the CleanTax system supports metadata curators in the task of creating, modifying and analyzing articulations between taxonomies, merging taxonomies, and merging taxonomically aligned datasets. CleanTax is also a platform for studying reasoning about taxonomies and articulations.

The CleanTax system takes as input any number of taxonomies, articulations between taxonomies, and sets of additional taxonomic assumptions, and answers the questions asked in section 1.4. CleanTax is extensible such that different reasoning engines may be plugged in. This means that CleanTax contains modules for converting inputs into formats understood by the given reasoners.

### 8.2.1 Input Formats

CleanTax supports several input formats:

#### Taxonomic Concept Schema

CleanTax accepts as input an XML document adhering to the Taxonomic Concept Schema (TCS). More information about TCS may be found at http://tdwg.napier.ac.uk/index.php?pagename=HomePage and the specification for the schema is here: http://tdwg.napier.ac.uk/doc/tdwg_tcs.pdf

Depending on the options set, the TCS document will be parsed into one or several CTI files, described below.

#### CleanTax Input

CleanTax also has a native format for inputting taxonomies and articulations. This format is described in Appendix A.3.1.

### 8.2.2 Output Formats

The primary product of a CleanTax run is a file describing the outcome of all the logic runs. The file can either be tab delimited or appear as an HTML table. If the latter, then hyperlinks to the input and output files of the reasoners are available, as well as a link to a graphic representation depicting inferred relations between nodes in the taxonomies. The reports provided by CleanTax are described in Appendix A.3.2.

## 8.3 Python

The command-line version of CleanTax supports TCS and CTI inputs and gives users control over what kind of logic tests to perform (implication, satisfiablity), which nodes to perform tests on, which relations to test, and which GTCs to try. If the input file is a TCS document, command-line options can select which taxonomies in the document should be used and can select specific species within the document to study. A full list of the command line options is provided in Appendix A.3.3.

In addition to the command line option, CleanTax may be used via a Web interface (see Figure 8.1). The interface supports reading taxonomic alignment files in the CTI format, applying single GTC sets, and it provides a variety of visualization options for the alignments. The Web interface also allows users to pick their favorite reasoner, or to permit CleanTax to determine the best reasoner for the given problem. In future work, the Web interface should support dataset integration. The expected result of a dataset integration may be seen in Figure 8.2. This output shows the alignment used to merge the datasets, the result of merging the aligned taxonomies, the starting datasets, and the resulting set of merged datasets. Before reaching this screen, the user is told how many possible merged datasets exist, and the user can opt not to display the merged datasets if there are too many. In the future, additional navigation of the possible merged datasets will be developed.

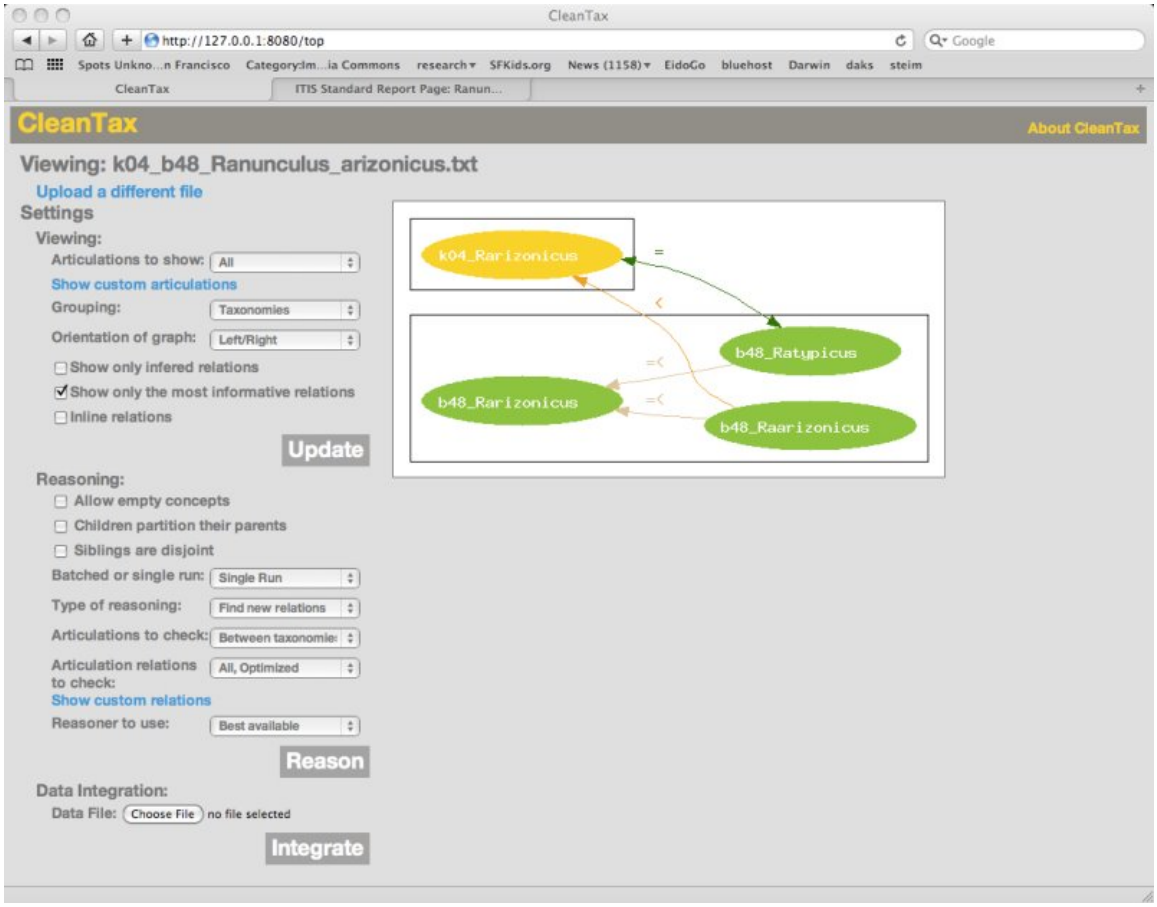



Figure 8.1: CleanTax Web interface

## 8.4 Workflows

Scientific workflows are networks of data analysis components linked together to perform complex tasks, including retrieving data from databases, performing specific algorithms on data, transforming data, and storing data. Scientific workflow environments, like Kepler [kep], allow researchers to construct and execute workflows on their desktop computers. This ability helps automate repetitive tasks and allows others to see how an analysis was performed.

### 8.4.1 Rationale for Workflow Implementation

Many aspects of the CleanTax system lend themselves to application within a workflow environment. For example, a Kepler actor that takes as input a taxonomy and a stream of global taxonomic constraint tokens could act like a filter, outputting only those constraints under which the taxonomy is consistent. Similarly, the reasoners used by CleanTax could be wrapped into generic "prover" and "model finder" actors, allowing people interested in benchmarking different reasoners to change the reasoner used in a given CleanTax workflow.

### 8.4.2 Functional Representation of CleanTax

Implementing CleanTax in Kepler is in a very early stage. The first step in the implementation has been to design functional programming signatures for the primary CleanTax functions. Workflows tend to be dataflow oriented, and there is a close relationship between dataflow-oriented systems and functional programming [LAB+06].

### 8.4.3 Basic Entities

Chapters 1 and 2 introduced many of the objects that are used throughout this dissertation. These *atomic* objects may be considered as opaque strings.

**type Authority = String** The name of an authority who stated something.

**type Node = String** The name of a node in a taxonomy.

**type Axiom = String** Axioms are precisely defined and could be subdivided into terms, connectives, and quantifiers. However, at this level of precision, axioms will be considered opaque strings.

**type Explanation = String** An explanation (*e.g.*, for a step in a proof).

**type Model = String** Similarly, a model, which provides an example of how a set of axioms might be satisfied, is a complex object, but at this level of precision, a model may be treated as an opaque string.

**type Relation = String** A symbol relating two nodes. Note that nodes are considered to be sets. In some formalizations, sets might be combined using set operations like intersection or union. In these cases, the results of the set operations will also be considered nodes for this level of formalization.

**type GTC = String** The name of a global taxonomic constraint meant to be held by all nodes in a taxonomy. These GTCs, when applied to a taxonomy represented by a given language, will result in additional axioms in that language.

**type TaxonomyRule = String** A rule defining a relationship between nodes in a taxonomy where the relationship is a partial order relationship. As before, this rule could be defined in a more fine-grained way. However, at this level of precision, it is best to consider this rule a simple string.

### 8.4.4 List Types

Some lists of the basic entities above are used frequently enough to deserve their own types.

**type CompoundGTC = [ GTC ]** Global taxonomic constraints can be grouped together (*e.g.*, non-emptiness and sibling disjointness). These compounds are often used as inputs, and can themselves be grouped into lists.

**type CompoundRelation = [ Relation ]** Single relations, such as equals, can be grouped into lists that are generally considered to indicate disjunctions of relations. For example, the list [ equals, disjoint ] represents the relation "equals or disjoint."

### 8.4.5 Complex Types

The following types are built up from the previously listed atomic types and other complex types.

**type ProofLine = (Axiom, Explanation)** A line of a proof is an axiom and a way that axiom was derived.

**type Proof = [ProofLine]** A proof is a list of proof lines.

**type ProofResult = (Boolean, Proof)** The result of an implication test is a true/false answer and a proof of that answer.

**type ConsistencyResult = (Boolean, Model)** The result of a consistency test is a true (found a model) / false (failed to find a model) answer, and if a model is found, the model.

**type Taxon = (Node, Authority)** A taxon is a node in a taxonomy and the authority who defined that taxon. Note that the authority for a taxon and the authority for the taxonomy containing that taxon may be different.

**type TestPair = (Taxon, Taxon)** We often want to compare taxa, testing whether or not some relationship holds between them.

**type TaxonomyConstraint = (Authority, Axiom)** A rule defining an additional constraint between nodes in a single taxonomy. An authority is needed because a constraint might be stated by the creator of the taxonomy, or an assumption made while analyzing the taxonomy.

**type Taxonomy = (Authority, [TaxonomyRule], [TaxonomyConstraint] )** A taxonomy has an authority, a list of rules defining that taxonomy, and a list of additional constraints.

**type Articulation = (Authority, Axiom)** A rule defining an inter-taxonomy constraint.

## 8.5 Basic Operations

Defined here are basic operations used to satisfy various use cases. The operations are described and given Haskell-style signatures. These operations may be seen as the basic commands in a language. They will be combined in the subsequent section to create more complex operations.

**powerset::** $[\,] \rightarrow [[\,]]$

Given a list of objects, the powerset function returns a list of lists where each sublist is a subset of the elements in the original list.

**proof::** $[Axiom] \rightarrow Axiom \rightarrow ProofResult$

Given a list of axioms and a goal, test whether the goal is implied by the axioms. Formally, does $[Axiom] \cup Goal \models \square$ ?

**consistency::** $[Axiom] \rightarrow ConsistencyResult$

Given a list of axioms, check whether the axioms are logically consistent. Formally, is there an $\mathcal{I}$ such that $\mathcal{I} \models [Axiom]$ ?

**axiomitizeTaxonomy::** $Taxonomy \rightarrow [Axiom]$

Render a given taxonomy into a set of axioms.

**axiomitizeArticulations::** $[Articulation] \rightarrow [Axiom]$

Render a set of articulations into a set of axioms.

**instantiateGTC::** $Taxonomy \rightarrow GTC \rightarrow [Axiom]$

Given a taxonomy and the name of a global taxonomic constraint, apply the GTC to the taxonomy and return a set of axioms that implement that GTC for that taxonomy.

**instantiateGTC::** $Taxonomy \rightarrow CompoundGTC \rightarrow [Axiom]$

Same as above, but apply a set of GTCs.

**instantiateGTC::** $[Articulation] \rightarrow GTC \rightarrow [Axiom]$

Given a set of articulations and the name of a global taxonomic constraint, apply the GTC to the articulations and return a set of axioms that implement that GTC for those articulations.

**instantiateGTC::** $[Articulation] \rightarrow CompoundGTC \rightarrow [Axiom]$

Same as above, but apply a set of GTCs.

**proved::** $ProofResult \rightarrow Boolean$

Given a proof result, return true if the goal was proved and false otherwise.

**consistent::** $ConsistencyResult \rightarrow Boolean$

Given a consistency result, return true if a model was found.

## 8.6 Complex Operations

The following operations are built up from the simple ones described in the previous section. Some of the operations use functional programming functions such as *filter*, *map*, *intersection*, and *union*. For more information about these functions, see [Jon02].

**consistentGTCSets::** $Taxonomy \rightarrow [CompoundGTC] \rightarrow [CompoundGTC]$

Given a taxonomy and a list of sets of GTCs (*e.g.*, [[ sibling disjointness, non-emptiness ], [ coverage, sibling disjointness ]]) return a list of GTC sets under which the taxonomy is consistent.

```
consistentGTCSets taxonomy gtcset = filter (consistent)
    (map (consistency (\gtc -> instantiateGTC taxonomy gtc) gtcset))
```

**consistentGTCSets::** $[Articulation] \rightarrow [CompoundGTC] \rightarrow [CompoundGTC]$

Given a taxonomy and a list of sets of GTCs (*e.g.*, [[ sibling disjointness, non-emptiness ], [ coverage, sibling disjointness ]]) return a list of GTC sets under which the taxonomy is consistent.

```
consistentGTCSets articulation gtcset = filter (consistent)
    (map (consistency (\gtc -> instantiateGTC taxonomy gtc) gtcset))
```

**allConsistent::** $Taxonomy \rightarrow Taxonomy \rightarrow [Articulation] \rightarrow [CompoundGTC] \rightarrow [CompoundGTC]$

Given two taxonomies, a set of articulations between them, and a list of sets of GTCs, return a list of GTC sets under which the taxonomies and articulations are consistent.

```
allConsistent t1 t2 arts gtcs =
  (intersection (consistentGTCSets t1 gtcs)
        (intersection (consistentGTCSets t2 gtcs) (consistentGTCSets arts gtcs)))
```

**testPairForRelation::** $Taxonomy \rightarrow Taxonomy \rightarrow [Articulation] \rightarrow [GTC] \rightarrow TestPair \rightarrow CompoundRelation \rightarrow ProofResult$

Given two taxonomies, a set of articulations between them, a compound GTC, a pair of nodes to test, and a relation to check, try to prove that the relation holds between the two nodes.

```
testPairForRelation t1 t2 arts gtc testpair relation =
  proof axioms goal
    where axioms = (union (instantiateGTC t1 gtc)
                          (union (instantiateGTC t2 gtc)
                          (instantiateGTC arts gtc)))
          goal = makeAxiom fst(testpair) snd(testpair) relation
```

**deductiveClosure::** Taxonomy → Taxonomy → [ Articulation ] → [ CompoundGTC ] → [ Node ] → [ Node ] → CompoundRelation → [ ProofResult ]

This computes the deductive closure of a pair of taxonomies and a set of articulations, given a set of relations to test and a set of assumptions under which to test them. Given two taxonomies, a set of articulations between them, a set of compound GTCs, two sets of nodes, and a set of relations to check, get every pair of nodes (one from

each of the provided groups) and check each relation in the list of relations under each compound GTC in that list.

```
deductiveClosure t1 t2 art gtcs nodeset1 nodeset2 relations =
 [testPairForRelation t1 t2 art gtc testpair relation |
      gtc <-- allConsistent t1 t2 art gtcs,
      testpair <-- getTestpairs t1 t2 nodeset1 nodeset2,
      relation <-- relations ]
```

## 8.7 Contributions and Future Work

This section described some of the details in implementing the CleanTax framework. A fair amount of progress has been made in building a traditional object-oriented version of the framework that supports pluggable reasoners and a variety of inputs formats and generates a number of useful reports. These reports include information about taxonomy and articulation consistency, calculations of an alignment's deductive closure, and statistics reflecting how long various processes took. Work on a Grid-enabled version of the framework has also been undertaken, although algorithms for effective distribution of tasks across multiple CPUs have yet to be written. Work on a scientific workflow version of CleanTax is at a very early stage, but functional descriptions of CleanTax features have been designed, which should simplify a migration of CleanTax to a system like Kepler.

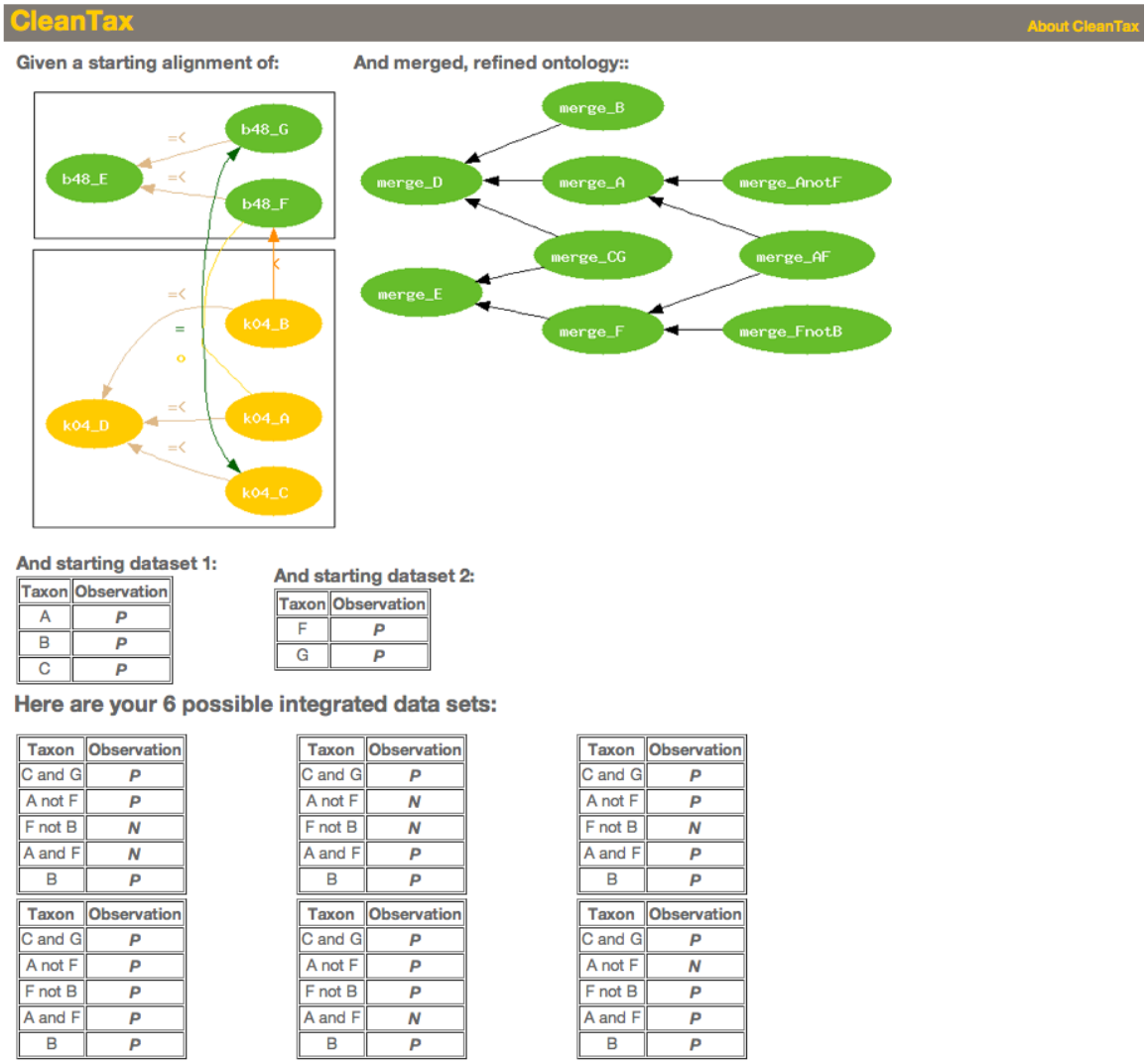



Figure 8.2: Results of a dataset merge in the CleanTax Web interface (notional).

# Chapter 9

# Possible Extensions: Explanations, Repairs, and Uncertainty

## 9.1 Overview and Objectives

Chapters 4 and 5 described mechanisms for discovering inconsistencies in taxonomies and articulations, and ways to discover new relationships between taxa. In an interactive setting, a metadata curator may want to know why a certain articulation was inferred, or why an inconsistency occurs. With these explanations in hand, a curator may want to repair an inconsistency, or if an inference is incorrect, discover ways to correct the inference while introducing no new inconsistencies or more incorrect inferences. Ambiguity should be avoided as well. The representation of taxonomies and articulations given so far provides precise ways of describing uncertainty. In the worst case, the relationship between two taxa is completely unknown, or maximally uncertain. In the best cases, two taxa are related with a single relation, such as "equals." In between there are cases with some uncertainty, such as "overlaps or disjoint." If the goal of a metadata curator is to provide the most specific metadata possible, tools for reducing uncertainty in taxonomies and articulations will be important. This chapter covers these three topics: explanations (section 9.2), repairs

(section 9.3), and reducing uncertainty (section 9.4).

## 9.2 Explanations

Users of automated reasoning systems often demand explanations for why a system reached a certain conclusion [BMPSB99]. Early work on expert systems, such as the MYCIN system for medical diagnosis [SDA+75], showed that users will not trust a system unless that system can explain its recommendations. A system like CleanTax, which judges alignments inconsistent or recommends additional articulations, must be able to explain its reasoning. This section begins with a brief survey of recent relevant explanation systems. It then describes some requirements an explanation system should meet. The section will end with a description of the currently implemented explanation system and plans for its extension.

### 9.2.1 Prior Work

Not many alignment systems implement an explanation component. Some exceptions to this are iMAP [DLD+04], a system for generating database schema matches, and work using Description Logics [MB95, McG96, BFH+99, DHS05].

iMAP uses a variety of search techniques to find one-to-one matches between elements in database schemas, as well as more complex matches, such as `address = concat(city, state)`. To help users interact with the system, iMAP employs an explanation module that can explain why a certain match was or was not made, and can provide reasons for the rankings it gives mappings. The explanations it offers often present the techniques iMAP used to make a decision, as well as dependencies that may have prevented iMAP from selecting a given match. For example, in a real-estate setting, an attempt to match `list-price` in one schema to `price * (1 + monthly-fee-rate)` in another schema might fail because `monthly-fee-rate` has already been matched to `month-posted`, which is known to be different from `list-price`. Most of the explanation module of iMAP depends on

this type of dependency, reducing its relevance to the CleanTax framework somewhat. However, the requirements presented in the iMAP work are relevant. [DLD+04] lists three types of questions that users ask in an alignment task: explain an existing match, explain an absent match, and explain match ranking. All three of these types of questions are relevant in the CleanTax framework. The latter becomes sensible when attempting to reduce uncertainty (see section 9.4).

In Description Logic, all reasoning can be reduced to questions about concept subsumption. The classic texts on explanations of reasoning in DL [MB95, McG96, BFH+99] provide a set of explanation rules for describing subsumption reasoning using a sequent calculus that parallels the tableaux reasoning used in DL. Examples of these rules are "Apply the double negation elimination rule" and "Apply de Morgan's laws." These explanations work well for those familiar with logic, but are not suitable for naïve users. [DHS05] argues that $\mathcal{L}_{\text{FOL}}$ resolution can help shed light on explanations for DL reasoning. They cite research from [Eis91] showing that any $\mathcal{L}_{\text{FOL}}$ resolution proof has a minimal refutation graph. This minimal refutation graph includes only those formulas contributing to the proof. The framework described by [DHS05] translates DL axioms into $\mathcal{L}_{\text{FOL}}$ for the purpose of generating simple explanations for the unsatisfiability of a given DL concept. As the CleanTax framework already employes varieties of $\mathcal{L}_{\text{FOL}}$, minimal refutation graphs will form an important part of the CleanTax explanation module.

### 9.2.2 Requirements

Here is a brief list of desiderata for an explanation system.

#### Queries

As mentioned above, an explanation system should be able to explain why a taxonomy, articulation, or alignment is inconsistent. It should also be able to explain why an inference was made, as well as why an inference was *not* made. Finally, when providing ranked information, the system should be able to justify the ranking.

### Provenance

The provenance of information used in the reasoning is extremely important. In terms of a single proof, this includes the sources of the initial axioms used in the proof. Provenance information comprises both the way the information was introduced into the system, as well as the original source of information itself. For example, a biological taxon has an authority, such as Linnaeus, 1754, and it also plays a role in a provided taxonomy, Kartesz, 2004. Similarly, an articulation may be stated by an expert (Peet, 2005) or it may be created by the system.

### Completeness and Clarity

Explanations can range from complete proofs output by an automatic prover, to digested explanations meant for those unfamiliar with logic-based representations.

### Presentation Methods

An explanation system should support both textual explanations and graphical representations of explanations. The former lend themselves to portability in terms of how an explanation system can be integrated into other systems. The latter can improve the clarity of the connections between various aspects of an explanation.

### Navigating

The explanation of one relationship often raises questions about other relationships. If an automatic reasoner deduces something surprising involving one taxon in a taxonomy, there may be other surprising relations involving that taxon. Therefore, the ability to navigate explanations is important. Navigating an explanation allows users to ask followup questions.

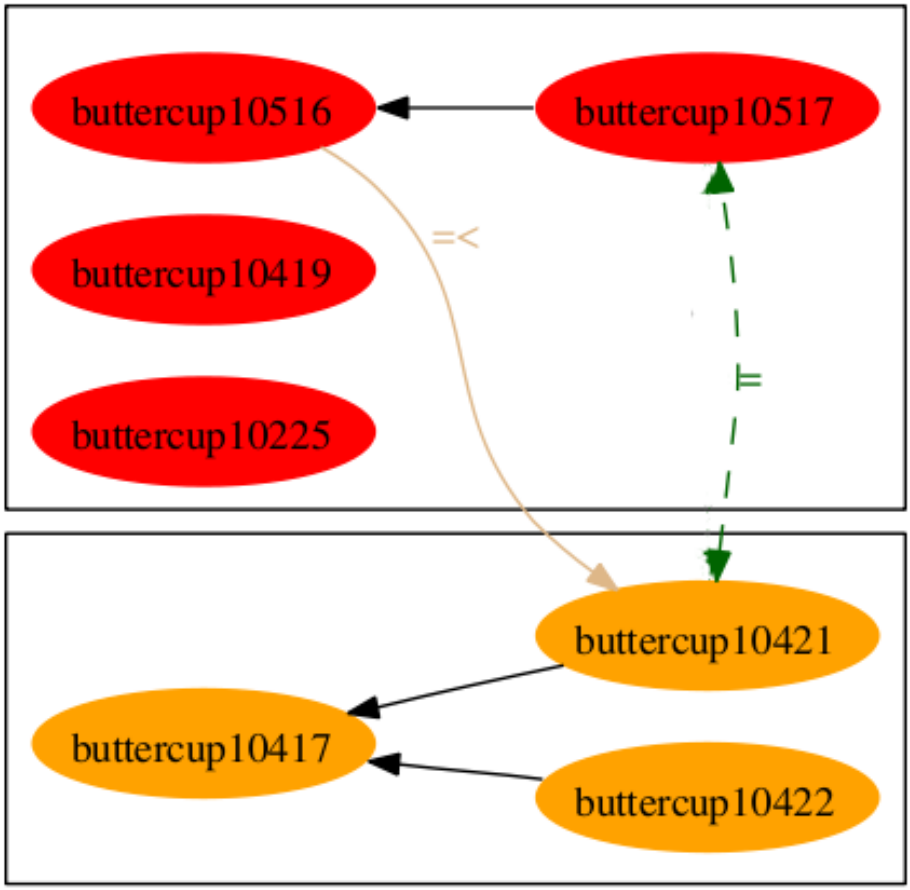



Figure 9.1: Representing only a subset of the $\mathbb{R}_{32}$ relations in an alignment. The dashed line represents an inferred $\equiv$ relation.

### 9.2.3 Initial Results

Initial work on the explanation module for CleanTax has focussed on using graphviz [gra] to sensibly render images of alignments and Prover9 proof trees. This work was done in collaboration with two UC Davis undergraduate students: Sichen Bao and Sean Riddle.

### Drawing Alignments

The simplest interface to an explanation for an alignment is a graphical representation of the alignment itself. Given that every pair of nodes in an alignment has *some* relation between them, the graph can be quite cluttered. One way to limit the clutter is to draw only a subset of the $\mathbb{R}_{32}$ relations. For example, Figure 9.1 shows only $\leq_{isa}$ relations, and the $\equiv$ and $\{\equiv,\subsetneq\}$ articulations. The dashed line between the `buttercup10517` taxon and the `buttercup10421` taxon represents an inferred relation. Another way to reduce the clutter in alignment graphs is to omit obvious existing relations, notably reflexive relations ($N \equiv N$) and transitive edges (if $N \subsetneq M$ and $M \subsetneq L$, the transitive relation $N \subsetneq L$ is not drawn). Initial experiments with designing interactive interfaces for clearly visualizing alignments have had some success, but far more work is necessary in this area.

### Drawing and Navigating Proof Trees

Clicking on an edge in Figure 9.1 should result in some sort of explanation. The edge may represent a relation supplied by an expert, or it may represent an inference. If it is an inference, the proof resulting in that relation should be shown. Figure 9.2 shows an initial attempt at representing a proof. This proof tree is the result of running a PROVER9 proof through a program that ships with PROVER9 called `prooftrans`, and then running that output through a program by David A Wheeler called `gvizify` [gvi] to convert the proof into a `dot` file, which is then drawn using the graphviz dot program. This proof explains the derivation of the articulation *buttercup*10517 $\supsetneq$ *buttercup*10521. The double octagon represents the goal, and single octagons represent clauses created by denying the goal (the articulation being tested). Ovals represent resolutions steps, and steps with no shapes around them represent clausification. Perhaps the most important part of the graph are the square nodes, which represent the formulas in the alignment used to derive the proof.

Each of the clauses in Figure 9.2 may appear in many other proofs. It may be useful for a user to click on a node and retrieve the other proofs that used that clause. Creating

links between proofs in this way can help identify interesting or problematic aspects of the alignment.

### 9.2.4 Future Work

The proof tree shown in Figure 9.2 is legible to those who understand logic and the process of resolution. However, it is far from a generally useful explanation tool. Creating visualizations that will be informative to a general user remains work to be done. In addition, the creation of textual explanations based on these proof trees remains to be done. The restricted domain of taxonomies and articulations may make building a visual and textual explanation system simpler than it is for proofs in a more general first-order language. Another task for future work is to build a mechanism for querying the alignment graph about articulations it did *not* make. The failure to prove an articulation cannot be represented using a proof tree (after all, the proof failed). Instead, the best way to explain the *lack* of an articulation may be to present a consistent interpretation $\mathcal{I}$ under which the articulation is violated. The MACE4 model finder can find such models, but representing the models is not straightforward. Future work will involve building an interface to ask questions about missing articulations, and mechanisms for displaying consistent models that violate the articulation in question.

## 9.3 Repairs

Alignments may be inconsistent, or if they are consistent, they may contain inferences deemed incorrect by an expert. In both of these cases, the alignment must be repaired in some way. The cases are quite different. In the former case, a prover can be used to pinpoint the formulas that lead to the discovered inconsistencies (see above discussion on minimal refutation graphs). The latter case is trickier, and relates strongly to the topic of belief revision, which will be dealt with in the section on uncertainty (section 9.4).

### 9.3.1 Prior Work

Much of the work on repairing inconsistencies in databases [Ger96, MT99, GGZ03] focuses on repairing violations of integrity constraints caused by the introduction of tuples. An exception to this rule is [Har01], which focuses specifically on inconsistencies in constraints. However, the algorithms described in [Har01] do not consider logic-based constraints, but rather work from a graph-theoretic perspective. Even with these domain mismatches, the work on inconsistency repair in databases introduces a number of terms and ideas that will arise later in this section.

Inconsistencies in ontologies have received more attention. [HvHH+05] presents four different contexts for inconsistencies in ontologies: (i) ensuring that new information does not make a consistent ontology inconsistent, (ii) diagnosing and repairing extant inconsistencies, (iii) attempting to reason despite the inconsistencies, and (iv) managing inconsistencies between different versions of the same ontology. Inconsistency in the CleanTax framework is primarily of the second kind.

In addition to defining types of inconsistencies, [HvHH+05] provides definitions for several important concepts, such as *maximal consistent sub-ontology* and *minimal consistent sub-ontology*. These definitions transfer easily to the domain of taxonomies.

**Definition 9.1** (Maximal consistent sub-taxonomy)**.** A taxonomy, $T'$ is a *maximal consistent sub-taxonomy* of $T$ if $T' \subseteq T$ and $T'$ is consistent and every $T''$ with $T' \subset T'' \subseteq T$ is inconsistent.

**Definition 9.2** (Minimal inconsistent sub-taxonomy)**.** A taxonomy $T'$ is a minimal inconsistent sub-taxonomy of $T$ if $T' \subseteq T$ and $T'$ is inconsistent and every $T''$ with $T'' \subset T'$ is consistent.

Finally [HvHH+05] provides algorithms for locating inconsistencies and repairing them once they have been located. The algorithms first find an unsatisfiable concept, and then find the *minimal sub-ontology* for that concept. This is the subset of the ontology in which

the concept is unsatisfiable, minus any redundant formulas. Once this subset is discovered, the minimal set of axioms that need to be removed is calculated (though [HvHH+05] do not provide an algorithm for this).

A similar approach is taken by the DION [SH05] prototype. Rather than present a general framework for ontology repair, as in [HvHH+05], DION is very closely tied to reasoning in a Description Logic framework. However, DION has the ability to pinpoint concepts in ontologies that contribute to many local inconsistencies in an ontology. Focusing on these problematic concepts may result in a faster repair time for the entire ontology. DION provides several algorithms for pinpointing inconsistencies, and [SH05] shows that their complexity is exponential to the size of the set of formulas contributing to inconsistencies.

### 9.3.2 Future Work

Currently, no effort has been made to repair inconsistencies in CleanTax. In the future, algorithms such as those outlined in [SH05] should be incorporated. Because the CleanTax framework is more focused than that of Description Logic ontologies, there is a good chance that more efficient algorithms for pinpointing multiple inconsistencies and deriving the minimal sets of repairs will be found. The notions of global taxonomic constraints (GTCs) should also be applied when locating and repairing inconsistencies. As mentioned in 4.3.2, globally applied taxonomic constraints can be locally defeasible. Searching for local violations of a GTC might provide guidance for heuristics searching for minimal sets of repairs.

## 9.4 Uncertainty

Uncertainty is most frequently modeled by assigning each formula a probability or some other number representing the certainty that the formula is correct [Goo61]. This type of numerical approach casts reasoning under uncertainty into the worlds of Bayesian analysis and fuzzy logic [Zad65]. In MoReTaX [GB03b], uncertain knowledge is marked with a

question mark — a coarse boolean qualification rather than a probabilistic one. In this dissertation, on the other hand, uncertainty is modeled by disjunctive relations, such as "equals, includes, or overlaps." If the goal of a metadata curator is to create articulations with the least uncertainty, alignments with a great number of disjunctions should be avoided. This section describes how uncertainty is measured in the current context and ways to visualize uncertainty in articulated taxonomies.

### 9.4.1 Uncertainty Metrics

There are many ways to quantify the uncertainty in a set of taxonomies and articulations between them. The factors to consider when generating an uncertainty metric are:

**Taxon or relation centered.** Uncertainty may be calculated by considering the average uncertainty of the taxa, or an average uncertainty of the $\mathbb{R}_{32}$ relations between them.

**Taxonomic relation weights.** Each relation will be given a level of uncertainty. The most basic assignment is to set the weight of a taxonomic relation equal to the number of $\mathbb{B}_5$ relations in the disjunction. For example, the equals relation would receive a weight of 1, while complete uncertainty $\{\equiv,\subsetneq,\supsetneq,\oplus,!\}$ would receive a weight of 5. However, alternative mappings may be sensible. For example, a curator might consider the `isa` relation $\{\equiv,\subsetneq\}$ as one having no uncertainty and assign it a weight of 1.

**Taxonomic relation type weights.** There are several classes of taxonomic relations: the $\leq_{isa}$ constraints, additional intra-taxonomy constraints $\mathbf{T_C}$, and articulations $A_C$. Each of these classes of taxonomic relations might be assigned a different weight (potentially zero) when calculating the total uncertainty in an alignment.

**Source weights.** Taxa and relations might be provided by trusted sources, they might be posited by a metadata curator, or they might be inferred by an automatic reasoner. Different sources might be given different weights. For example, an intra-taxonomic

relation provided directly by the creator of a taxonomy might not be considered in the uncertainty calculation at all.

**Fine grained fixing.** A metadata curator might select, by hand, specific articulations or intra-taxonomy relations that may not be changed. These *inviolate* relations will not enter into a calculation of uncertainty.

These factors might be combined in a number of ways. For example:

**Average taxon uncertainty.** Each taxon is involved in a number of taxonomic relations. Sum the number of relations involved, giving a value of one to each $\mathbb{B}_5$ relation (so that complete uncertainty has a value of 5 and an `isa` relation has a value of 2) and divide by the number of relations. Sum up this value for each taxon and divide by the number of taxa.

**Average articulated taxon uncertainty.** Similar to above, but only include articulations when calculating taxon uncertainty.

**Average taxonomic relation uncertainty.** Each relation has a degree of uncertainty. Measure the average of these degrees.

**Average articulation uncertainty.** Similar to above, but only include articulations.

Each of these options recommends a different model for calculating and minimizing uncertainty. Each model transforms the goal of minimizing uncertainty in a given articulation into a search through a minimization space. This search may be addressed using traditional algorithms, such as A*. The exploration of these models awaits future work.

### 9.4.2 Reducing Uncertainty

Once a metric of uncertainty is in hand, the next question is how to reduce that uncertainty. In an interactive setting, a user should be presented with a total uncertainty, then a summary of the sources of that uncertainty. The type feedback presented to the user can

take various forms. The simplest and most general feedback would simply present a list of articulations in order of their uncertainty. Another type of feedback might involve the concepts in order of the average uncertainty in their incident relations.

A ranked listing of *decisions* that could be made to reduce uncertainty would also be useful. However, determining the impact of a change to an alignment is quite complex. Changing any articulation may alter the deductive closure of the alignment, creating potential inconsistencies, new articulations, and potentially more uncertainty. Any change must (i) leave unchanged those relations that have been marked as inviolate, and (ii) result in a consistent alignment.

The act of reducing uncertainty in an alignment places the problem squarely into the realm of belief change. In essence, changing a $\mathbb{R}_{32}$ constraint by removing a disjunction (*e.g.*, changing $\{\equiv,\subsetneq\}$ to $\{\equiv,\}$ may be seen as *belief revision* — adding information (not $\subsetneq$) while maintaining consistency. Similarly, changing a $\mathbb{R}_{32}$ constraint by adding a disjunction may be seen as *belief contraction* — the removal of a belief while maintaining consistency. The operations of revision and contraction have been studied extensively. In particular, the AGM framework [AGM85] describes a set of postulates defining a rational belief revision operator, a set of postulates defining a rational belief contraction operator, and shows how they are related. Within the AGM framework, notions of revision preference orderings [KM91] and iterated revision [DP97], can inform the order in which changes to an alignment can be made to minimize uncertainty while maximizing the plausibility of the resulting alignment.

### Minimal Uncertainty

A given alignment, along with an uncertainty metric, including a set of inviolate relations, will have a *minimum uncertainty*, defined as the consistent alignment with smallest possible uncertainty given the supplied uncertainty metric. Although this number may be calculated by doing a complete analysis of every possible combination of $\mathbb{R}_{32}$ relations on the malleable constraints, it is not certain that the minimum can be reached through iterative revision,

or even if it is sensible. It may, however, provide an interesting statistic for a user.

#### Legal Transformations and Reachability

Another factor in belief revision is the type of revisions that are legal. For example, given an $\mathbb{R}_{32}$ constraint, can a single revision change only one disjunction (either by adding one or removing one) or can a revision perform a more radical change? Decisions such as these can affect the "reachability" of possible worlds, where each alignment is considered a possible world. This touches on an alternative approach to belief revision, one that works on the models that satisfy a given set of formulas. [Dal88]

#### Heuristics

Belief revision can occur as a series of iterative steps, altering one constraint, then another. This iterative process defines a search space of transformations. This search space can be enormous, necessitating heuristics and optimizations for searching through the minimization space. These heuristics and optimizations may be used to rank decisions a user might make to reduce uncertainty. Some example heuristics are:

• Taxa at different "heights" in a taxonomy might have different impacts on uncertainty. For example, uncertainty in taxa with many upper bounds may propagate up the taxonomy.
• Some relations, or combinations of relations, should be removed as quickly as possible. For example, we have seen that removing all $\mathbb{R}_{32}$ relations containing $\{\subsetneq,\supsetneq\}$ except those containing $\oplus$ may allow the system to drop to a polynomial-time reasoning language.
• Some relations may be easier to remove. Again any combination containing $\{\subsetneq,\supsetneq\}$ might be easy to disambiguate.

### 9.4.3 Using Dataset Merges to Guide Uncertainty Reduction

The number of possible unambiguous dataset merge results is determined by the provided taxonomy alignments. If some of the possible dataset merges are known by the user to be

impossible, marking them as such may have implications for the articulations that led to the inclusion of those possible worlds. In other words, it may be possible to improve an alignment by eliminating possible merged datasets. It may also be possible to determine which articulations in the alignments engender the greatest number of possible worlds. Providing a user with a ranked list of articulations that created the biggest increases in the number of possible worlds can help improve the taxonomic alignments, reduce the number of possible merges, and increase the specificity of a single best-effort merge.

### 9.4.4 Visualizations

To assist a user in gaining a sense for where uncertainty lies, there should be "uncertainty views" of an articulation. These views can emphasize the uncertainty of taxa by emphasizing those that have a high average incident uncertainty, or emphasize the uncertainty in the constraints by using lines of different thickness for differing levels of uncertainty.

## 9.5 Additional Research and Development

**Supporting incremental changes to alignments.** The current CleanTax implementation rechecks the entire alignment whenever it checks for consistency and new inferences. We would like to be able to perform these checks incrementally, as users add, modify, or remove articulations in the alignments, or modify the taxonomies. The goal is to improve system response time for users creating and editing mappings between large taxonomies, which is frequently the case for many context domains (including species classifications).

**Merging and reasoning with more complex observational data.** Prior work described how to merge taxonomies [TBL08] and how to merge datasets when the data were simple presence/absence observations [TBL09b]. We would like to *expand the types* of data that may be merged in the system, starting with categorical data and numerical measures such as species abundance counts. We will also explore *context constraints* and their influence on the complexity of our merge algorithms. Context is used in many scien-

tific datasets to define the scope of particular observations. For instance, species may be observed under different environmental factors such as nitrogen treatments. In this case, the treatment attribute (the level of nitrogen treatment) denotes a context for the observed species, implying in this case that the individual observed in one treatment was different from an individual of the same type in another treatment context.

## 9.6 Conclusion

The contributions of this dissertation include a formalization of taxonomies, articulations, and alignments, as well as definitions of taxonomy and alignment consistency. Basing the articulation language on the RCC-5 topological algebra has proven useful in representing incomplete knowledge, and its application in the context of taxonomic alignment represents a new contribution. A representation of this formalization in monadic first-order logic has been presented, and optimizations for quickly calculating all relationships implied by an alignment have been given and evaluated. In addition, this dissertation has described a mechanism for merging taxonomies and merging datasets of taxonomically organized data. Work on creating a framework for testing algorithms and reasoners has resulted in a system that has been applied to real-world datasets and has discovered several inconsistencies and many new relationships. In addition, initial steps toward formalizing uncertainty reduction and building tools to support explanations of inferred relations have been taken and reported. Finally, preliminary research on repairing inconsistencies and applying other logics, such as Description Logics and propositional logic, has been undertaken and described.

Taxonomies exist all around us. Hopefully this thesis has shed some light on what these taxonomies mean, and how they relate to one another.

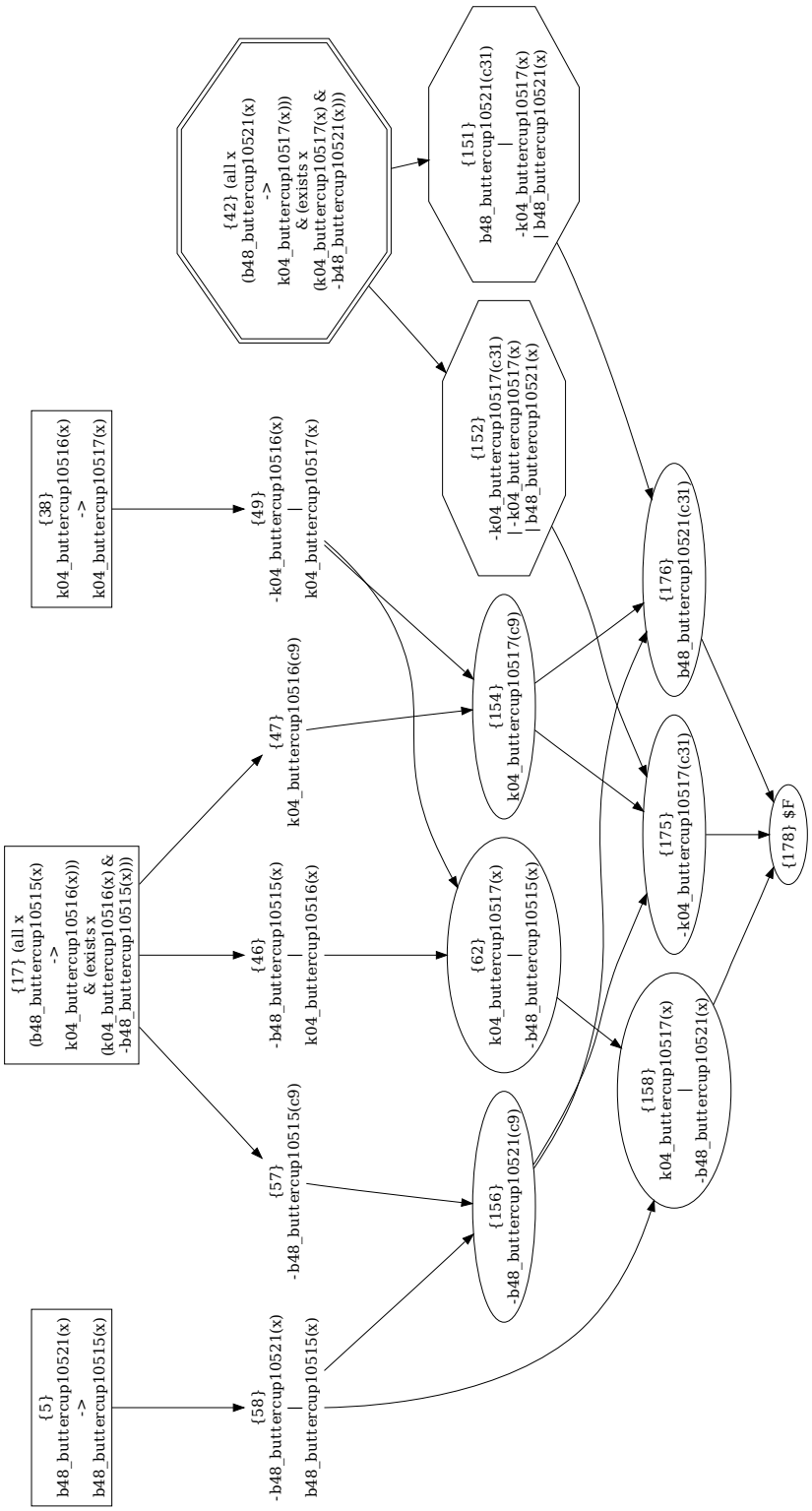



Figure 9.2: Graphical representation of the proof that *buttercup*10517 $\supsetneq$ *buttercup*10521 .

# Bibliography

[AGM85] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985. 153

[AGU72] Alfred V. Aho, M. R. Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972. 92, 94

[AGY05] A. Avesani, F. Giunchiglia, and M. Y. Yatskevich. A large taxonomy mapping evaluation. In *Proc. of the 4th International Semantic Web Conference (ISWC)*, pages 67–81, 2005. 2

[AJKO08] Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, pages 983–992. IEEE, 2008. 104

[AKG87] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. In Umeshwar Dayal and Irving L. Traiger, editors, *SIGMOD*, pages 34–48. ACM Press, 1987. 103

[AKO07] Lyublena Antova, Christoph Koch, and Dan Olteanu. World-set decompositions: expressiveness and efficient algorithms. In *Proceedings of ICDT*, pages 194–208, 2007. 108

[AL08] Marcelo Arenas and Leonid Libkin. XML data exchange: consistency and query answering. *Journal of the ACM*, 55(2):1–72, 2008. 121

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983. 77

[BBG01] Massimo Benerecetti, Paolo Bouquet, and Chiara Ghidini. *On the dimensions of context dependence: partiality, approximation, and perspective*, volume 2116, page 59. Springer, Berlin / Heidelberg, 2001. 9

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. 82, 177

[Beh22] Heinrich Behmann. Beiträge zur algebra der logik, insbesondere zum entscheidungsproblem. *Mathematische Annalen*, 86(3-4):163–229, September 1922. 37

[Ben48] L. D. Benson. A treatise on the North American Ranunculi. *American Midland Naturalist*, 40:1–261, 1948. 49

[Ben94] Brandon Bennett. Spatial reasoning with propositional logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR'94: Principles of Knowledge Representation and Reasoning*, pages 51–62. Morgan Kaufmann, San Francisco, California, 1994. xii, 40, 76, 77

[Ber95] Walter G. Berendsohn. The concept of "potential taxa" in databases. *Taxon*, 44:207–212, 1995. 35

[Ber03] Walter G. Berendsohn. *MoReTax – Handling Factual Information Linked to Taxonomic Concepts in Biology*. Number 39 in Schriftenreihe für Vegetationskunde. Bundesamt für Naturschutz, 2003. 35, 163, 164

[BFH+99] Alexander Borgida, Enrico Franconi, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Explaining ALC subsumption. In Patrick Lambrix, Alexander Borgida, Maurizio Lenzerini, Ralf Möller, and Peter F. Patel-Schneider, editors, *Description Logics*, volume 22 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1999. 143, 144

[BGW93] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Set constraints are the monadic class. In *Logic in Computer Science*, pages 75–83, 1993. 37, 73, 82, 113

[BJBM01] Chad Berkley, Matthew Jones, Jivka Bojilova, and Daniel Higgins Metacat. Metacat: a schema-independent XML database system. In *SSDBM*, pages 171–179, 2001. 98, 121

[Blu07] Stan Blum. Stan Blum, personal communication. Personal Communication, January 2007. 46

[BMPSB99] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, and Alexander Borgida. Reducing CLASSIC to practice: knowledge representation theory meets reality. *Artificial Intelligence*, 114(1-2):203–237, 1999. 143

[BMS08] Shawn Bowers, Joshua Madin, and Mark Schildhauer. A conceptual modeling framework for expressing observational data semantics. *Conceptual Modeling - ER 2008*, pages 41–54, 2008. 106

[BMSZ03] Paolo Bouquet, B. Magnini, L. Serafini, and S. Zanobini. A SAT-based algorithm for context matching. In *IV International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'2003)*, Stanford University (CA, USA), June 2003. 34

[BPB93] J. H. Beach, S. Pramanik, and J. H. Beaman. Hierarchic taxonomic databases. In R. Fortuner, editor, *Advances in Computer Methods for Systematic Biology:*

*Artificial Intelligence, Databases, Computer Vision*, chapter 15, pages 241–256. Johns Hopkins University Press, Baltimore, 1993. 35

[Bra83] R.J. Brachman. What IS-A is and isn't: an analysis of taxonomic links in semantic networks. *IEEE Computer*, 16:30–36, 1983. 8, 20, 101

[CdQ06] P. D. Cantino and K. de Queiroz. Phylocode: a phylogenetic code of biological nomenclature. http://www.ohio.edu/phylocode/, 2006. 9

[CGL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tailoring OWL for data intensive ontologies. In *Proceedings of the OWL: Experiences and Directions Workshop, Galway, Ireland*, 2005. 82

[Cha05] Arthur D. Chapman. Principles of data quality. Technical report, Global Biodiversity Information Facility, Copenhagen, 2005. 102

[CHSR09] Andrew D. Cliff, Peter Haggett, and Matthew Smallman-Raynor. The changing shape of island epidemics: historical trends in icelandic infectious disease waves, 1902-1988. *Journal of Historical Geography*, 35(3):545–567, 2009. 98

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936. 37, 82

[Cle91] James F. Clement. *Birds of the World: A Checklist*. Ibis Publishing Co., 4th edition, 1991. ix, 4

[Cle01] James F. Clement. *Birds of the World: A Checklist*. Ibis Publishing Co., 5th edition, 2001. ix, 4

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium*, pages 151–158, New York, 1971. ACM. 82

[Dal88] Mukesh Dalal. Investigations into a theory of knowledge base revision. In *AAAI*, pages 475–479, 1988. 154

[Dar04] Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 328–332. IOS Press, 2004. 120

[DHS05] Xi Deng, Volker Haarslev, and Nematollaah Shiri. A framework for explaining reasoning in description logics. In Thomas Roth-Berghofer and Stefan Schulz, editors, *ExaCt*, volume FS-05-04 of *AAAI Technical Report*, pages 55–61. AAAI Press, 2005. 143, 144

[DLD+04] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Y. Halevy, and Pedro Domingos. iMAP: discovering complex mappings between database schemas. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *SIGMOD Conference*, pages 383–394. ACM, 2004. 143, 144

[DLNN97] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997. 73

[DMQ05] D. Dou, D. McDermott, and P. Qi. Ontology translation on the semantic web. In *Journal on Data Semantics (JoDS)*, volume II, pages 35–57, 2005. 34, 83, 84, 85, 88, 92

[DP97] Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89(1–2):1–29, January 1997. 153

[DP02] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, second edition, 2002. 15

[DR02] Hong Hai Do and Erhard Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. Morgan Kaufmann, 2002. 11

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, second edition, 1994. 72, 174

[Ehr07] Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap*, volume 4 of *Semantic Web And Beyond Computing for Human Experience*. Springer, 2007. 20, 23, 33, 35, 48

[Eis91] N. Eisinger. *Completeness, Confluence, and Related Properties of Clause Graph Resolution*. Morgan Kaufmann, 1991. 144

[Euz04] J. Euzenat. State of the art on ontology alignment. http://www.starlab.vub.ac.be/publications/kweb-223.pdf, 2004. 23, 33

[EV04] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-Lite. In R. López de Mántaras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pages 333–337. IOS Press, 2004. 74

[FKP03] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. In *PODS*, pages 90–101. ACM, 2003. 121

[FKPT07] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Quasi-inverses of schema mappings. In Leonid Libkin, editor, *PODS*, pages 123–132. ACM, 2007. 121, 125

[FPW07] Nico M. Franz, Robert K. Peet, and Alan S. Weakley. On the use of taxonomic concepts in support of biodiversity research and taxonomy. In Quentin D. Wheeler, editor, *The New Taxonomy, Systematics Association Special Volume Series 74*, pages 61–84. Taylor and Francis, Boca Raton, FL., 2007. 41, 83

[GB03a] M. Geoffroy and W. G. Berendsohn. The concept problem in taxonomy: importance, compontents, approaches. [Ber03], pages 5–14. 41

[GB03b] Marc Geoffroy and Walter G. Berendsohn. Transmission of taxon-related factual information. [Ber03], pages 83–86. 150

[GBM07] R. Grutter and B. Bauer-Messmer. Towards spatial reasoning in the semantic web: a hybrid knowledge representation system architecture. In *Lecture Notes in Geoinformation and Cartography*, Berlin, Heidelberg, 2007. Springer. 75

[Gen01] Wolfgang Gentzsch. Sun Grid Engine: towards creating a compute power grid. In *CCGRID*, pages 35–39. IEEE Computer Society, 2001. 81

[Ger96] Michael Gertz. An extensible framework for repairing constraint violations. In *Workshop on Foundations of Models and Languages for Data and Objects*, pages 41–56, 1996. 149

[GG03] M. Geoffroy and A. Güntsch. Assembling and navigating the potential taxon graph. [Ber03], pages 71–82. 36

[GGZ03] Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1389–1408, 2003. 149

[Goo61] I. J. Good. A causal calculus. *British Journal of the Philosophy of Science*, 11:305–318, 1961. 150

[gra] Graphviz. http://www.graphviz.org/. 146

[GSY04] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *Proc. of the First European Semantic Web Symposium - ESWS*, pages 61–75, 10-12 May 2004. 34, 76

[gvi] Gvizify Python script to convert Prover9 proofs to dot format. http://www.dwheeler.com/formal_methods/gvizify. 147

[GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Heidelberg, 1999. 9

[GW02] Nicola Guarino and Christopher A. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002. 20

[GYS07] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *J. Data Semantics*, 9:1–38, 2007. 11

[Har01] Sven Hartmann. Coping with inconsistent constraint specifications. In Hideko S. Kunii, Sushil Jajodia, and Arne Sølvberg, editors, *ER*, volume 2224 of *Lecture Notes in Computer Science*, pages 241–255. Springer, 2001. 149

[Hor07] Ian Horrocks. Ian Horricks, personal communication. Personal Communication, November 2007. 75

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In *Proceedings of the Fourth International Semantic Web Conference*, pages 353–367, 2005. 149, 150

[IR88] Y. E. Ioannidis and R. Ramakrishnan. An efficient transitive closure algorithm. In *Proceedings of the 14th International Conference Very Large Databases*, pages 382–394, Los Angeles, California, August 1988. 94

[JD97] Peter Jonsson and Thomas Drakengren. A complete classification of tractability in RCC-5. *Journal of Artificial Intelligence Research*, 6:211–221, 1997. 40, 78, 82

[Jon02] Simon Peyton Jones, editor. *Haskell 98 Language and Libraries: The Revised Report*. http://haskell.org/, September 2002. 138

[Kar04] John T. Kartesz. Synthesis of North American flora. BONAP, North Carolina Botanical Garden, 2004. 49

[kep] Kepler: A system for scientific workflows. http://kepler-project.org. 133

[KG05] Yarden Katz and Bernardo Cuenca Grau. B.C.: representing qualitative spatial information in OWL-DL. In *In: Proceedings of OWL: Experiences and Directions*, 2005. 74, 75

[KJH+05] Jaehong Kim, Minsu Jang, Young-Guk Ha, Joo-Chan Sohn, and Sang-Jo Lee. MoA: OWL ontology merging and alignment tool for the semantic web. In Moonis Ali and Floriana Esposito, editors, *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE*, volume 3533 of *Lecture Notes in Computer Science*, pages 722–731. Springer, 2005. 74, 84

[KKP05] J. Kennedy, R. Kukla, and T. Paterson. Scientific names are ambiguous as identifiers for biological taxa: their context and definition are required for accurate data integration. In *International Workshop on Data Integration in the Life Sciences (DILS)*, LNCS 3615, pages 80–95, July 2005. 2

[KM91] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *KR*, pages 387–394, 1991. 153

[Kor08] K. Korovin. iProver – an instantiation-based theorem prover for first-order logic (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *IJCAR 2008*, pages 292–298. Springer, 2008. 80, 120

[KS03] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review Journal*, 18(1):1–31, 2003. 33

[KSBG00] M. Koperski, M. Sauer, W. Braun, and S.R. Gradstein. *Referenzliste der Moose Deutschlands*, volume 34. Schriftenreihe für Vegetationskunde, 2000. 83

[KV04] Konstantinos Kotis and George A. Vouros. The HCONE approach to ontology merging. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *Proceedings of the First European Semantic Web Symposium*, volume 3053 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2004. 83

[KVS06] Konstantinos Kotis, George A. Vouros, and Konstantinos Stergiou. Towards automatic merging of domain ontologies: the HCONE-merge approach. *Journal of Web Semantics*, 4(1):60–79, 2006. 83

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006. 4, 134

[LB87] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence Journal*, 3:78–93, 1987. 73

[Len02] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGART Symposium on Principles of Database*, 2002. 127

[LL59] C.I. Lewis and C.H. Langford. *Symbolic Logic*. New York: Dover, 2nd edition, 1959. 103

[MB95] Deborah L. McGuinness and Alexander Borgida. Explaining subsumption in description logics. In *IJCAI (1)*, pages 816–821, 1995. 143, 144

[McD98] Daniel McDermott. PDDL - the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998. 34

[McG96] Deborah L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers, State University of New Jersey, 1996. 143, 144

[MD02] Drew V. McDermott and Dejing Dou. Representing disjunction and quantifiers in RDF. In Ian Horrocks and James A. Hendler, editors, *International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2002. 34

[MFRW00a] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *ECAI-04*, Breckenridge, Colorado, April 2000. 85, 128

[MFRW00b] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. The Chimaera ontology environment. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 1123–1124. AAAI Press / The MIT Press, 2000. 85

[MHH+01] Renée J. Miller, Mauricio A. Hernández, Laura M. Haas, Ling-Ling Yan, C. T. Howard Ho, Ronald Fagin, and Lucian Popa. The Clio project: managing heterogeneity. *SIGMOD Record*, 30(1):78–83, 2001. 12, 33

[MT99] Enric Mayol and Ernest Teniente. A survey of current methods for integrity constraint maintenance and view updating. In Peter P. Chen, David W. Embley, Jacques Kouloumdjian, Stephen W. Liddle, and John F. Roddick, editors, *ER (Workshops)*, volume 1727 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 1999. 149

[MW82] David Makowski and Rolf M. Wulfsberg. An improved taxonomy of postsecndary institutions. Technical report, National Center for Higher Education Management Systems, P.O. Drawer P, Boulder Colorad 80302, 1982. 2

[MWJ99] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proceedings of the 2nd International Conference On Information FUSION'99*, 1999. 22

[MWK00] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. *Lecture Notes in Computer Science*, 1777:86–100, 2000. 22, 47

[Neb95] Bernhard Nebel. Computational properties of qualitative spatial reasoning: first results. In Ipke Wachsmuth, Claus-Rainer Rollinger, and Wilfried Brauer, editors, *KI*, volume 981 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 1995. 82

[NM00] Natalya Fridman Noy and Mark A. Musen. PROMPT: algorithm and tool for automated ontology merging and alignment. In *AAAI/IAAI*, pages 450–455, 2000. 128

[NM03] Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003. 11, 33, 83, 85, 88

[Pee05] Robert K. Peet. Taxonomic concept mappings for 9 taxonomies of the genus Ranunculus published from 1948 to 2004. Unpublished dataset., June 2005. 1, 48, 121

[PTS06] Duc Pham, John Thornton, and Abdul Sattar. Towards an efficient SAT encoding for temporal reasoning. *Principles and Practice of Constraint Programming - CP 2006*, pages 421–436, 2006. 76

[PTU03] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. Dike: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. *Software: Practice and Experience*, 33(9):847–884, 2003. 35

[que03] Map LC (LCC) to Dewey (DDC) classification, 2003. 2

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001. 33

[RCC92] David A. Randell, Zhan Cui, and Anthony Cohn. A spatial logic based on regions and connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92*, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992. 40

[RN99] Jochen Renz and Bernhard Nebel. On the complexity of qualitative spatial reasoning: a maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1-2):69–123, 1999. 78

[RV02] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Communications*, 15(2-3):91–110, 2002. 82

[SDA+75] E.H. Shortliffe, R. Davis, S.G. Axline, B.G. Buchanan, C.C. Green, and S.N. Cohen. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the MYCIN system. *Computers and Biomedical Research*, 8:303–320, 1975. 143

[SE07] Pavel Shvaido and Jérôme Euzenat. *Ontology Matching*. Springer, Heidelberg, 2007. 33

[SH05] Stefan Schlobach and Zhisheng Huang. Inconsistent ontology diagnosis: framework and prototype. Technical report, 2005. 150

[SM01a] G. Stumme and A. Maedche. FCA-MERGE: bottom-up merging of ontologies. In *Proceedings of the 17$^{th}$ International Joint Conference on Artificial Intelligence*, pages 225–234, 2001. 11, 84

[SM01b] G. Stumme and A. Maedche. Ontology merging for federated ontologies on the semantic web. In *Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001)*, pages 413–418, 2001. 83, 128

[SS09] Markus Stocker and Evren Sirin. Pelletspatial: a hybrid RCC-8 and RD-F/OWL reasoning and query engine. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *Proceedings of OWL:Experiences and Directions*, 2009. 75

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 1(48):1–26, 1991. 73, 74

[SSW05] H. Stuckenschmidt, L. Serafini, and H. Wache. Reasoning about ontology mappings. Technical report, ITC-IRST, Trento, 2005. 33

[ST05] Luciano Serafini and Andrei Tamilin. DRAGO: distributed reasoning architecture for the semantic web. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2005. 11, 33

[TBL08] David Thau, Shawn Bowers, and Bertram Ludäscher. Merging taxonomies under RCC-5 algebraic articulations. In *CIKM Workshop on Ontologies and Information Systems for the Semantic Web (ONISW)*, pages 47–54. ACM, 2008. 83, 155

[TBL09a] David Thau, Shawn Bowers, and Bertram Ludäscher. Cleantax: a framework for reasoning about taxonomies. In *Proceedings of the AAAI Spring Symposium*, pages 49–50, 2009. 63

[TBL09b] David Thau, Shawn Bowers, and Bertram Ludäscher. Merging sets of taxonomically organized data using concept mappings under uncertainty. In Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors, *OTM Conferences*, volume 5871 of *Lecture Notes in Computer Science*, pages 1103–1120. Springer, 2009. 83, 98, 155

[TBL10] David Thau, Shawn Bowers, and Bertram Ludäscher. Towards best-effort merge of taxonomically organized data. In *Proceedings of the 2nd International Workshop on New Trends in Information Integration (in conjunction with ICDE)*, 2010. 98

[TCS] The taxonomic concept schema. http://tdwg.napier.ac.uk/. 35, 40, 41, 48

[tdw] Taxonomic data working group. http://www.tdwg.org/. 35, 40

[Tha08] David Thau. Reasoning about taxonomies and articulations. In *Ph.D. '08: Proceedings of the 2008 EDBT Ph.D. workshop*, pages 11–19, New York, NY, USA, 2008. ACM. 33, 63

[TL07] David Thau and Bertram Ludäscher. Reasoning about taxonomies in first-order logic. *Ecological Informatics*, 2(3):195–209, 2007. 33

[Tur36] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. In *Procedings of the London Mathematical Society*, volume 42 of *2*, pages 230–265, 1936. 37, 82

[VMP05] Pascal Vasseur, El Mustapha Mouaddib, and Claude Pégard. Introduction to multisensor data fusion. In Richard Zurawski, editor, *The Industrial Information Technology Handbook*, pages 1–10. CRC Press, 2005. 128

[wik07] Comparison of dewey and library of congress subject classification, November 2007. 2

[W.W08] W.W.McCune. Prover 9: http://www.cs.unm.edu/ mccune/prover9/, July 2008. 56

[WW09] Matthias Westphal and Stefan Wölfl. Qualitative CSP, finite CSP, and SAT: comparing methods for qualitative constraint-based reasoning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 628–633, 2009. 77

[WWG09] Matthias Westphal, Stefan Wölfl, and Zeno Gantner. GQR: a fast solver for binary qualitative constraint networks. In *AAAI Spring Symposium on Benchmarking of Qualitative Spatial and Temporal Reasoning Systems*, 2009. 77, 80

[xbr] eXtensible Business Reporting Language. http://www.xbrl.org/. 7

[Zad65] Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. 150

# Appendix A

# Formal Languages and Proofs

## A.1 Formal Languages

### A.1.1 First-Order Logic (FOL)

For ease of reference, we summarize basic notions from first-order predicate logic [EFT94].

**Syntax.** The language $\mathcal{L}_{\text{FOL}}$ of first-order logic is built from an *alphabet* consisting of (i) a set of *variables* $\mathbf{V} = \{x, y, z, \dots\}$, (ii) *connectives* $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (*not*, *and*, *or*, *if-then*, *if-and-only-if*), (iii) *quantifiers* $\forall, \exists$ (*for all* and *there exists*), and (iv) a *signature* $\mathbf{S} = \mathbf{R} \cup \mathbf{F} \cup \mathbf{C}$, involving sets of *relation symbols* $\mathbf{R}$ $(R_1, R_2, \dots)$, *function symbols* $\mathbf{F}$ $(f_1, f_2, \dots)$, and *constants* $\mathbf{C}$ $(c_1, c_2, \dots)$. Each $R \in \mathbf{R}$ and $f \in \mathbf{F}$ has a unique *arity* $\geq 1$.

The set $\mathbf{T}$ of *terms* is the least set such that (i) $\mathbf{V}, \mathbf{C} \subseteq \mathbf{T}$ (constants and variables are terms), and (ii) for every $k$-ary $f \in \mathbf{F}$ and $t_1, \dots, t_k \in \mathbf{T}$, also $f(t_1, \dots, t_k) \in \mathbf{T}$.

A $\mathcal{L}_{\text{FOL}}$ *formula* is either an *atomic formula* $R(t_1, \dots, t_k)$, with $k$-ary $R \in \mathbf{R}$ and $t_1, \dots, t_k \in \mathbf{T}$, or of the form $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$, $(\neg\varphi)$, $\forall x(\varphi)$, or $\exists x(\varphi)$, where $\varphi, \psi$ are $\mathcal{L}_{\text{FOL}}$ formulas, and $x \in \mathbf{V}$. Parentheses may be omitted when clear from the context.

**Semantics.** Fix a signature $\mathbf{S} = \mathbf{R} \cup \mathbf{F} \cup \mathbf{C}$. A first-order *structure* $\mathcal{I} = (D, I)$ for $\mathbf{S}$ consists of a *domain* $D$ and a mapping $I$, assigning to every constant $c \in \mathbf{C}$, $k$-ary function symbol $f \in \mathbf{F}$, and $k$-ary relation symbol $R \in \mathbf{R}$, a domain element $c^I$, a $k$-ary function $f^I : D^k \to D$, and a $k$-ary relation $R^I \subseteq D^k$, respectively. Since $I$ interprets (*i.e.*, assigns meaning to) all symbols in $\mathbf{S}$, terms and formulas over $\mathbf{S}$ can be *evaluated* under $\mathcal{I}$, provided we also map free variables to domain elements via a *variable assignment* $\beta : \mathbf{V} \to D$. Let $\mathfrak{I} = (\mathcal{I}, \beta)$ be an *interpretation*, *i.e.*, a first-order structure $\mathcal{I}$ with variable assignment $\beta$. Formula evaluation is defined inductively as a *satisfaction relation* $\mathfrak{I} \models \varphi$ ("$\mathfrak{I}$ *satisfies* $\varphi$", "$\mathfrak{I}$ *is a model of* $\varphi$", "$\varphi$ *holds in* $\mathfrak{I}$"):[1]

$$
\begin{array}{lll}
\mathfrak{I} \models R(t_1, \ldots, t_n) & \text{:iff} & (\mathfrak{I}(t_1), \ldots, \mathfrak{I}(t_n)) \in R^I \\
\mathfrak{I} \models \neg\varphi & \text{:iff} & \text{not } \mathfrak{I} \models \varphi \\
\mathfrak{I} \models \varphi \wedge \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ and } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \vee \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ or } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \to \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ implies } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \leftrightarrow \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ if and only if } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \forall x(\varphi) & \text{:iff} & \mathfrak{I} \models \varphi \text{ for all } d \in D \text{ holds } \mathfrak{I}\frac{d}{x} \models \varphi \\
\mathfrak{I} \models \exists x(\varphi) & \text{:iff} & \mathfrak{I} \models \varphi \text{ there exists } d \in D \text{ such that } \mathfrak{I}\frac{d}{x} \models \varphi
\end{array}
$$

These abstract notions become more tangible when recast in database terminology: A formula $\varphi(x_1, \ldots, x_n)$ over $\mathbf{S}$ with free variables[2] $x_1, \ldots, x_n$ defines an $n$-ary *query* $q$ over the *schema* $\mathbf{S}$. In particular, checking for which tuples $(c_1, \ldots, c_n)$ we have $\mathcal{I} \models \varphi(c_1, \ldots, c_n)$ is exactly the same as running the query $q$ against the *database instance* $\mathcal{I}$, *i.e.*, $q(\mathcal{I}) = \{ (c_1, \ldots, c_n) \mid \mathcal{I} \models \varphi(c_1, \ldots, c_n) \}$. Here, we use $\mathcal{I}$ instead of $\mathfrak{I}$, since all free variables $x_i$ of $\varphi$ have been substituted by constants $c_i$ (so $\beta$ is not needed).

[1] As usual, 'iff' means 'if and only if'. The colon ':' indicates that the left-hand side is defined by the right-hand side. $\mathfrak{I}\frac{d}{x}$ is the same as $\mathfrak{I}$, but $\beta$ is modified to map $x$ to $d$, *i.e.*, $\beta(x) := d$.

[2] An occurrence of a variable $x$ in $\varphi$ is *free* if it is *not* under the scope of a quantifier $\forall x(\cdots)$ or $\exists x(\cdots)$; else it is called *bound*.

A formula $\varphi$ without free variables is called a *sentence* or *constraint*, and corresponds to a yes/no (*boolean*) query. Let $\Phi$ be a set of constraints. We write $\mathcal{I} \models \Phi$, if $\mathcal{I} \models \varphi$ for all $\varphi \in \Phi$ and say "$\mathcal{I}$ *is a model of* $\Phi$". We write $\Phi \models \varphi$ if every model of $\Phi$ is also a model of $\varphi$, *i.e.*, $\varphi$ is a (logical) *consequence* of $\Phi$.[3]

### A.1.2 The Syntax and Semantics of Monadic First-Order Logic

**Syntax.** The language of monadic first-order logic is built from an alphabet consisting of (i) a set of variables $\mathbf{V} = \{x, y, z, \dots\}$, (ii) connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (*not*, *and*, *or*, *if-then*, *if-and-only-if*), (iii) *quantifiers* $\forall, \exists$ (*for all* and *there exists*), and (iv) a signature, $\mathbf{S} = \mathbf{R} \cup \mathbf{C}$, involving sets of relation symbols $\mathbf{R}$ and constants $\mathbf{C}$.

A $\mathcal{L}_{\text{MFOL}}$ *formula* is either an *atomic formula* $R(t)$, where $t$ is either a variable or constant, or of the form $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$, $(\neg\varphi)$, $\forall x\colon \varphi$, or $\exists x\colon \varphi$, where $\varphi, \psi$ are $\mathcal{L}_{\text{MFOL}}$ formulas, and $x$ is a variable.

**Semantics.** Fix a signature $\mathbf{S} = \mathbf{R} \cup \mathbf{C}$. A first-order *structure* $\mathcal{I} = (D, I)$ for $\mathbf{S}$ consists of a *domain* $D$ and a mapping $I$, assigning to every constant $c \in \mathbf{C}$ and unary relation symbol $R \in \mathbf{R}$, a domain element $c^I$, and a unary relation $R^I \subseteq D$, respectively.

Since $I$ interprets (*i.e.*, assigns meaning to) all symbols in $\mathbf{S}$, terms and formulas over $\mathbf{S}$ can be *evaluated* under $\mathcal{I}$, provided we also map free variables to domain elements via a *variable assignment* $\beta : \mathbf{V} \rightarrow D$. Let $\mathfrak{I} = (\mathcal{I}, \beta)$ be an *interpretation*, *i.e.*, a first-order structure $\mathcal{I}$ with variable assignment $\beta$. Formula evaluation is defined inductively as a

[3] Note the difference between $\mathcal{I} \models \varphi$ and $\Phi \models \varphi$: the former is the satisfaction relation between a structure (database instance) $\mathcal{I}$ and a formula (query) $\varphi$; the latter is the consequence relation, stating that *all* structures $\mathcal{I}$ which satisfy $\Phi$, also satisfy $\varphi$. Thus, $\mathcal{I} \models \varphi$ is also called formula evaluation (given $\mathcal{I}$), while the $\Phi \models \varphi$ involves "reasoning" (independent of $\mathcal{I}$).

*satisfaction relation* $\mathfrak{I} \models \varphi$ ("$\mathfrak{I}$ *satisfies* $\varphi$", "$\mathfrak{I}$ *is a model of* $\varphi$", "$\varphi$ *holds in* $\mathfrak{I}$"):[4]

$$
\begin{array}{lll}
\mathfrak{I} \models R(t) & \text{:iff} & \mathfrak{I}(t) \in R^I \\
\mathfrak{I} \models \neg\varphi & \text{:iff} & \text{not } \mathfrak{I} \models \varphi \\
\mathfrak{I} \models \varphi \wedge \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ and } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \vee \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ or } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \rightarrow \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ implies } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \varphi \leftrightarrow \psi & \text{:iff} & \mathfrak{I} \models \varphi \text{ if and only if } \mathfrak{I} \models \psi \\
\mathfrak{I} \models \forall x(\varphi) & \text{:iff} & \mathfrak{I} \models \varphi \text{ for all } d \in D \text{ holds } \mathfrak{I}\frac{d}{x} \models \varphi \\
\mathfrak{I} \models \exists x(\varphi) & \text{:iff} & \mathfrak{I} \models \varphi \text{ there exists } d \in D \text{ such that } \mathfrak{I}\frac{d}{x} \models \varphi
\end{array}
$$

An *interpretation* $\mathcal{I}$ maps the symbols in a signature $\mathbf{S} = \mathbf{R} \cup \mathbf{C}$ to relations and objects in a modeled "real world." A formula $\varphi$ without free variables is called a *constraint* and corresponds to a yes/no (boolean) query. We use $\Phi$ to represent a set of such constraints. If an *interpretation* makes true all constraints in $\Phi$, we write $\mathcal{I} \models \Phi$, and say "$\mathcal{I}$ *satisfies* (or *is a model of*) $\Phi$". If a formula $\varphi$ is a logical consequence of a set of constraints[5] $\Phi$, we write $\Phi \models \varphi$. See Example 4.1 in Section 4.3.1 for a concrete example of interpretations and constraints.

### A.1.3 The Syntax and Semantics of $\mathcal{AL}$

The following definitions of the syntax and semantics of $\mathcal{AL}$ come mainly from [BCM+03].

**Syntax.** Concepts in $\mathcal{AL}$ are formed according to this rule:

C, D →

A | (atomic concept)

[4] As usual, 'iff' means 'if and only if'. The colon ':' indicates that the left-hand side is defined by the right-hand side. $\mathfrak{I}\frac{d}{x}$ is the same as $\mathfrak{I}$, but $\beta$ is modified to map $x$ to $d$, *i.e.*, $\beta(x) := d$.

[5] $\varphi$ is a logical consequence of $\Phi$ if every model of $\Phi$ is also a model of $\varphi$

$\top$ | (universal concept)

$\bot$ | (bottom concept)

$\neg$A | (atomic negation)

$C \sqcap D$ | (intersection)

$\forall R.C$ | (value restriction)

$\exists R.\top$ | (limited existential quantification)

**Semantics.** Interpretations $\mathcal{I}$ in $\mathcal{AL}$ consist of a non-empty set $\mathcal{U}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept $A$ a set $A^{\mathcal{I}} \subseteq \mathcal{U}^{\mathcal{I}}$ and to every atomic role $R$ a binary relation $R^{\mathcal{I}} \subseteq \mathcal{U}^{\mathcal{I}} \times \mathcal{U}^{\mathcal{I}}$. Concept descriptions are interpreted inductively as follows:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \mathcal{U}^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \mathcal{U}^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \mathcal{U}^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \mathcal{U}^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}
\end{aligned}
$$

## A.2 The Maximal Tractable Subalgebra $\mathbb{R}_5^{28}$

Table A.1 describes the 32 $\mathbb{R}_{32}$relations, and the $\mathbb{R}_5^{28}$ subset which leads to polynomial time reasoning.

## A.3 Implementation Details

### A.3.1 The CleanTax Input File

**Example A.1.** `taxonomy b48 Benson 1948`

| $\mathbb{R}_{32}$ **Relations** | $\mathbb{R}_5^{28}$ |
|---|---|
| $\{\ \}$ | • |
| $\{!\}$ | • |
| $\{\oplus\}$ | • |
| $\{!, \oplus\}$ | • |
| $\{\subsetneq\}$ | • |
| $\{!, \subsetneq\}$ | • |
| $\{\oplus, \subsetneq\}$ | • |
| $\{!, \oplus, \subsetneq\}$ | • |
| $\{\supsetneq\}$ | • |
| $\{!, \supsetneq\}$ | • |
| $\{\oplus, \supsetneq\}$ | • |
| $\{!, \oplus, \supsetneq\}$ | • |
| $\{\subsetneq, \supsetneq\}$ | |
| $\{!, \subsetneq, \supsetneq\}$ | |
| $\{\oplus, \subsetneq, \supsetneq\}$ | • |
| $\{!, \oplus, \subsetneq, \supsetneq\}$ | • |

| $\mathbb{R}_{32}$ **Relations** | $\mathbb{R}_5^{28}$ |
|---|---|
| $\{\equiv\}$ | • |
| $\{!, \equiv\}$ | • |
| $\{\oplus, \equiv\}$ | • |
| $\{!, \oplus, \equiv\}$ | • |
| $\{\subsetneq, \equiv\}$ | • |
| $\{!, \subsetneq, \equiv\}$ | • |
| $\{\oplus, \subsetneq, \equiv\}$ | • |
| $\{!, \oplus, \subsetneq, \equiv\}$ | • |
| $\{\supsetneq, \equiv\}$ | • |
| $\{!, \supsetneq, \equiv\}$ | • |
| $\{\oplus, \supsetneq, \equiv\}$ | • |
| $\{!, \oplus, \supsetneq, \equiv\}$ | • |
| $\{\subsetneq, \supsetneq, \equiv\}$ | |
| $\{!, \subsetneq, \supsetneq, \equiv\}$ | |
| $\{\oplus, \subsetneq, \supsetneq, \equiv\}$ | • |
| $\{!, \oplus, \subsetneq, \supsetneq, \equiv\}$ | • |

Table A.1: The maximal tractable subalgebra $\mathbb{R}_5^{28}$: only relations marked "•" are in $\mathbb{R}_5^{28}$.

```
(a b c)


taxonomy k04 Kartesz 2004

(a b c)


articulation p05 Peet 2005

[b48_a equals k04_a]

[b48_b equals k04_b]

[b48_c equals k04_c]


goal

<possible ? b48_a equals k04_c>
```

A taxaLogic input file is divided into sections with tree types of headers: taxonomy, articulation, and goal. The taxonomy and articulation headers are followed by an "authority", which is a string with no spaces. Any characters after the authority (excluding intervening spaces) are considered a "full name" of the authority. This is optional. In the example above, the first taxonomy has authority b48, the "full name" of which is "Benson 1948." Full names can have any alphanumeric characters and are for output formatting purposes.

There may be multiple of each kind of header. If there are multiple taxonomy or articulation headers, each must have a different authority string. There must be at least one taxonomy header. Case matters in the file. Blank lines are optional. The order of the sections is not mandatory, except that a node and authority must be declared before it is used in an articulation or a goal. It is traditional to declare the taxonomies first, followed by the articulations, and then a goal, if one exists.

Below each header, on the next line, is the information about that header. Each type of header has a different sort of information with a different format.

**Taxonomy headers.** Each line after a taxonomy header, and until the next header appears, contains information about a taxonomy. Each line represents a single node and its immediate children. The name of each node must have no spaces, underscores, or dashes. The nodes are *not* prepended with any authority information. A line under the taxonomy header looks like:

($parent\ child_1\ child_2\ child_3\ \ldots\ child_n$)

Note the parentheses around the entire line. As mentioned above, the first node is the parent, and every node after that is a child of the parent.

**Articulation header.** Each line in an articulation section is one articulation between two nodes. The format of the line is

[$authority1\_node1\ relations\ authority2\_node2$]

Note the square brackets around everything in the line. The first and third terms are nodes named in the taxonomy section, prepended with their respective authorities, with an underscore between the authority string and the node string. The authority and node strings must have been declared in a prior taxonomy section.

There are five relations: equals, includes, is_included_in, disjoint, overlaps.

If only one relation holds between the two nodes, it can be listed without further embellishment:

$[b48\_a\ equals\ k04\_a]$

If multiple relations may hold, those relations may appear in a space-separated list within curly braces:

$[b48\_a\ \{equals\ overlaps\}\ k04\_a]$

This signifies that either an equals or an overlaps relation holds between these two nodes. If the relation is completely unknown, then all the relations appear in the curly braces.

The relations may also be bounded by asterisks:

$[b48\_a\ *\{equals\ overlaps\}*\ k04\_a]$ or $[b48\_a\ *equals*\ k04\_a]$

This signifies that the relations were inferred by the CleanTax system rather than given by a curator. This is useful for displaying the results of a CleanTax run.

**The goal header.** Each line in a goal section contains a single goal. There are two types of goals: implies and possible. The "implies" goal tests to see if a given relationship between two nodes is a consequence of the taxonomies, articulations, and any additional rules that may be added at some point. The "possible" goal checks to see if the relationship being tested might possibly be true, or in other words, checks that it is not impossible that the relation holds. Each goal line appears between $<>$ signs:

$<possible\ ?\ b48\_a\ equals\ k04\_c>$

The first word is the type of goal, possible or implied, followed by a question mark, and then the relationship being tested. As in the articulation section, if a number of

relationships are of interest, they may be placed in a space separated list between curly braces:

$< possible\ ?\ b48\_a\ \{equals\ includes\}\ k04\_c >$

This checks to see if it is possible that either the equals and includes relationships holds between the two nodes.

The Extended Backus-Naur form for the CTI File appears in Figure A.1.

### A.3.2 Output Formats

#### Main report

The main report displays the following information: Whether or not each taxonomy + articulation + set of GTC is consistent. For any two nodes, and any R32 relation, whether or not that relation holds. Links to the input and output of the reasoners. Links to a graphical representation of the taxonomies and articulations with discovered articulations.

The columns of the output and the table are (examples in parentheses):

- the name of the run (k04_b48_Ranunculus_petiolaris)
- the short name of the first taxonomy (k04)
- the short name of the second taxonomy (b48)
- the GTC used for the run (NonEmptiness Coverage DisjointChildren)
- the type of run (consistency or implied or possible)
- whether the relation was given or inferred (given or inferred)
- the name of node 1 (buttercup10517)
- the name of node 2 (buttercup10222)
- the relation being tested (equals includes disjoint)
- the outcome (true, false, unclear)

```
taxaLogicInput = taxonomy-section, [{taxonomy-section}],
       [{articulation-section}], [{goal-section}];
taxonomy-section = taxonomy-header, <EOL>, [{node-descriptor}];
taxonomy-header = 'taxonomy', white-space, authority, white-space, phrase,
        <EOL>;
node-descriptor = '(', node, white-space, [{node | white-space}], ')',
       <EOL>;
articulation-section = articulation-header, <EOL>,
       [{articulation-descriptor}];
articulation-header = 'articulation', white-space, authority, white-space,
       phrase, <EOL>;
articulation-descriptor = '[', articulation, ']', <EOL>;
articuliation = authority, '_', node, white-space, relation-string,
       white-space, authority, '_', node;
relation-string = relation | '{', {relation | white-space}, '}';
relation = 'equals' | 'includes' | 'is_included_in' | 'disjoint' |
       'overlaps';
goal-section = goal-header, <EOL>, [{goal-descriptor}];
goal-header = 'goal';
goal-descriptor = '<',goal-type, white-space, '?', white-space,
       articulation, '>', <EOL>;
goal-type = 'implied' | 'possible'
node = word;
authority = word;
white-space = {? white space characters ?};
word-letter = alphanumeric - ('_','-', ? white space characters ?);
word = word-letter, [{word-letter}];
phrase = word, [{" " | word}];
```

Figure A.1: The Extended Backus-Naur form for the CTI File

- the name of, or a link to, the prover9 input file
- the name of, or a link to, the prover9 output file
- the name of, or a link to, the mace4 input file
- the name of, or a link to, the mace4 output file

**Timing report**

The outputs of reasoner files often contain information about the proofs and how long they took. The timing report lists, for each reasoner, for each proof, a single line of statistics. This report currently has only been created for prover9.

The report is currently generated by a Perl script named p9StatsSummary.pl. It takes as input the output of a Prover9 run.

The output columns are:

- run name
- node 1
- node 2
- relationship
- user time
- system time
- wall time
- proof steps

### Maximally Informative Relation

Many relations will hold between any two nodes. For example, if a equals b, then a {equals includes} b. There will be one distinguished maximally informative relation which is true and implies the truth of all the other true relations. This report determines the maximally informative relation for each pair of nodes.

The report is generated by a Perl script named findMaxInfNode.pl. It takes as input a tab-delimited output file from CleanTax and outputs a report with one row per implication test.

Its columns are:

- run name
- gtc
- node 1
- node 2
- maximally informative relation

### R32 Lattice

The R32 Lattice tracks how often each relation is true in a given run. There are three interesting numbers for each relation in the lattice: the number of times it was true, the number of times it was the maximally informative relation, and the number of times the relation was inferred to be true if a relation-test reducing optimization is performed.

This report is currently generated with a Perl script named inferLattice.pl. This script takes as input a tab delimited output file from CleanTax and outputs a report with a row for each relation; the columns are:

- a number representing the relation
- the relation

- the number of times the relation was determined to be true by the reasoners
- the number of times the relation was true according to the optimization
- the number of times the relation was the maximally informative relation

#### Generating a R32 Lattice for GraphViz

The R32 lattice can be visualized in GraphViz. Currently, a Perl script named relLattice-DoThreeVals.pl takes a file output by the R32 Lattice report and outputs a dot file which can be processed by GraphViz.

#### Generating a tab-delimited report from an HTML report

CleanTax can output a tab-delimited text file or an HTML table. If the output of a run is in HTML form, it can be converted to a tab-delimited file using a Perl script named tlo2csv.pl. This script is useful for creating files that may be consumed by the other report scripts described here.

### A.3.3 CleanTax Command-Line Options

CleanTax supports the following options, those with asterisks are not yet implemented :

#### General options

- -r : root directory
- -o : output file - if left out, goes to stdout
- -p : program directory - where the reasoners live
- -m : directory to store intermediate files
- -T : reasoner timeout in seconds

**Input options**

*CTI Input Options*

- -i : input single CTI file
- -d : input directory of CTI files

*Reasoner input options*

- -D : input directory of reasoner input files
- -I : input single reasoner file

*TCS schema input options*

- -x : xml file to parse into CTI files
- -s : species in the XML file, either a single species or a csv string
- -t : a list of authority/abbreviation tuples (*e.g.*, "[('Kartesz 2004','k04'),('Benson 1948','b48')")

*Program output options*

- -h : HTML table output

NOTE: One of -i, -d, or -x must be specified.

**Latent taxonomic assumption options**

Unless otherwise specified, no GTCs will be applied.

- -l : a csv string of GTCs to apply, for example:"n,nc,ncd,cd,none" will run GTC sets [non-empty], [non-empty and coverage], [non-empty, coverage, and sibling disjointness], [coverage and sibling disjointness], and no GTCs
- -v : a csv list of single GTCs, and run the power set. For example "n,c,d" will run the power set "n,c,d,nc,nd,cd,ncd and none"

**Goal options**

- -n : nodes to test, either"all": compare all pairwise nodes, or a set of tuples to test "(a,d) (b,e) (c,f)", * or two csv lists, one for T1 and one for T2 "a,b,c d,e,f",
- -c : relations to test - this is a csv list, such as: "equals,{overlaps equals},overlaps,is_included_in"
- -a : a power set of relations given in a csv list
- -w : test types, either "all", or a comma delimited string composed of consistent, implied and/or possible (*e.g.*, "consistent,implied". If excluded, no goals are tested (only consistency of axioms).

## A.4 Proofs

In the following we include several formal, automated proofs mentioned in the dissertation. We used two automatic reasoners: Prover9 and Mace4.[6] Prover9 is a resolution-based first-order logic theorem prover. Thus, in order to prove $\Phi \models \varphi$, *i.e.*, that a formula $\varphi$ follows from a set of assumptions (or *axioms*) $\Phi$, adds the negated formula $\neg\varphi$ as a goal to the assumptions, trying to refute the conjunction $\Phi \wedge \neg\varphi$, using (primarily) logic resolution steps: $\Phi \cup \{\neg\varphi\} \vdash \square$. Here, "$\square$" stands for the *empty clause*, denoting False, and "$\vdash$" denotes the provability relation, based on the legal derivation rules steps of the reasoning calculus at hand (*e.g.*, first-order resolution). If indeed, False can be derived, it follows from the correctness of the calculus that $\varphi$ is a logical consequence of $\Phi$.

If Prover9 fails to find a proof, this may be because (a) indeed $\varphi$ is *not* a consequence of $\Phi$, or (b) it is, but Prover9 simply could not find it (given the limited time and memory resources). One can then use the Mace4 tool to try and find models of $\Phi \wedge \neg\varphi$, *i.e.*, interpretations $\mathcal{I}$ such that $\mathcal{I} \models (\Phi \wedge \neg\varphi)$ holds. When successful, $\mathcal{I}$ is in fact a *counter model* for the desired theorem $\varphi$, demonstrating that all the assumptions $\Phi$ can be satisfied while still falsifying $\varphi$.

[6] http://www.cs.unm.edu/~mccune/mace4/

If MACE4 fails to find a model for $\Phi \wedge \neg\varphi$ after PROVER9 also failed to show that $\Phi \wedge \neg\varphi$ is inconsistent, then one cannot discern between the cases (a) and (b). In general, this situation can occur because the logical implication and satisfiability of FOL are undecidable problems. However, these questions *are* decidable for $\mathcal{L}_{\text{tax}}$ and—in the case of using only $\mathbb{R}_5^{28}$ constraints—even efficiently decidable in polynomial time.

Summarizing, PROVER9 and MACE4 were used in tandem when searching for proofs or for counter-models of $\Phi \models \varphi$, respectively.

### A.4.1 Automated Reasoning Examples for Figure 1.3

Figure 1.3 presented a number of questions about a pair of taxonomies and a mapping between them. We examined the effects of each of our latent taxonomic assumptions: *non-emptiness*, *sibling disjointness*, and *coverage* on each of the questions asked in the figure. Given the three GTCs, there are eight possible GTC combinations (*e.g.*, none of the GTCs, all of the GTCs, just coverage and non-emptiness, *etc.*)

**Figure 1.3c: Is C $\subseteq$ E implied?** We used PROVER9 to find proofs for each of the combinations of possible GTCs. When the hypothesis could not be proven, we used MACE4 to verify that there was a counter example. We found that the hypothesis could only be proven if the *coverage* GTC was applied. The other GTCs had no effect on the outcome. Below is the proof, in PROVER9 syntax, with only the coverage GTC applied. The correspondence between PROVER9 syntax and the syntax in Table 4.2 should be clear.

```
2     c(x) -> a(x)      Assumption
4     a(x) <-> d(x)     Assumption
7     d(x) -> e(x)      Assumption
8     c(x) -> e(x)      Goal
12    c(c1)             Deny 8
13    -c(x) | a(x)      Clausify 2
15    -d(x) | e(x)      Clausify 7
```

```
17    -e(c1)           Deny 8
19    -a(x) | d(x)     Clausify 4
20    a(c1)            Resolve 12 13
21    d(c1)            Resolve 20 19
22    -d(c1)           Resolve 17 15
23    False            Resolve 21 22
```

**Figure 1.3d: Is C ≡ D possible?** Here we used MACE4 to find models of the formulas under various GTC combinations. MACE4 found models for all combinations of GTCs unless *non-emptiness* and *sibling disjointness* were *both* assumed. PROVER9 could not prove that $C \equiv D$ does not follow from the other formulas. However, the reason that MACE4 could not find a model for this situation is clear when one looks at the figure: If $C \equiv D$, then $A$, $C$, and $D$ must be identical, *i.e.*, contain the same elements. If $B$ is non-empty, and $B$ and $C$ are disjoint, then $A$ must contain some element which is not in $C$. That contradicts the statement that $A$ and $C$ are equivalent.

**Figure 1.3e: Is A ≡ E possible?** We again used MACE4 to find models that satisfy this set of formulas under various combinations of GTCs. In this situation, the assertion *is* indeed possible regardless of whether or not any GTCs are asserted.

### A.4.2 Inconsistent Taxonomies and Mappings: Figure 4.5

The PROVER9 theorem prover shows that the non-emptiness, sibling disjointness, and coverage GTCs render the taxonomies and mappings in Figure 5 inconsistent. The proof posits an instance of *BRhn*. It then reasons that the instance cannot be one of *BRhs* or *BRht* because of the sibling disjointness constraint. The equivalence articulations from these nodes to their counterpart *KRhs* and *KRht* nodes imply that the instance of *BRhn* cannot be an instance of either of these. Through the coverage constraint, the instance cannot be an instance of *KRh*. Following the equivalence articulation to *BRh* means that the instance

cannot be in *BRh*. However, this leads to a contradiction because *BRhn* implies *BRh*.

```
1   BRhn(x) -> BRh(x)           Assumption
6   BRh(x) <-> KRh(x)           Assumption
7   BRhs(x) <-> KRhs(x)         Assumption
8   BRht(x) <-> KRht(x)         Assumption
10  (exists x (BRhn(x)))        Assumption
16  BRhn(x) -> -BRhs(x)         Assumption
17  BRhn(x) -> -BRht(x)         Assumption
21  KRh(x) -> KRhs(x) | KRht(x) Assumption
22  BRhn(c2)                    Clausify 10
23  -BRhn(x) | BRh(x)           Clausify 1
24  -BRhn(x) | -BRhs(x)         Clausify 16
25  -BRhn(x) | -BRht(x)         Clausify 17
27  BRhs(x) | -KRhs(x)          Clausify 7
32  -BRhs(c2)                   Resolve 24 22
33  BRht(x) | -KRht(x)          Clausify 8
37  -BRht(c2)                   Resolve 25 22
43  -KRh(x) | KRhs(x) | KRht(x) Clausify 21
46  -KRhs(c2)                   Resolve 32 27
53  -KRht(c2)                   Resolve 37 33
57  -KRh(c2) | KRht(c2)         Resolve 46 43
60  -BRh(x) | KRh(x)            Clausify 6
67  -KRh(c2)                    Resolve 57 53
69  BRh(c2)                     Resolve 22 23
79  -BRh(c2)                    Resolve 67 60
80  False                       Copy 79, unit_del 69
```