2010

Identifying and eliminating inconsistencies in mappings across hierarchical ontologies

Bhavesh Sanghvi
Iowa State University

Identifying and eliminating inconsistencies in mappings across hierarchical ontologies

by

Bhavesh Sanghvi

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Iowa State University

Ames, Iowa

2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The successful completion of this dissertation is the result of dedicated efforts put in by many people and this report will be incomplete without giving due credit to them. This acknowledgment is but a small token of gratitude in recognition of their help in my endeavor.

I am highly grateful to my major advisor Dr. Vasant Honavar for his encouragement to select this topic and for his guidance throughout the work on it. I thank him for the belief he showed in me and I hope to have come near the level of his expectations.

I am deeply thankful to my committee members Dr. Giora Slutzki and Dr. Samik Basu for their insightful comments and valuable feedback at various times and helping me during the course of this work. I am also thankful to them for agreeing to be on my program of study committee.

I also want to express my gratitude for all the members of the Artificial Intelligence laboratory, specially, Ganesh Ram Santhanam, Neeraj Kaul, and Fadi Towfic for helping me on various occasions. Their valuable suggestions and friendly attitude were really helpful.

I offer my regards to Linda Dutton and Lanette Woodard for their help and support throughout my graduate studies in the Department of Computer Science and research in the Artificial Intelligence laboratory. I also wish to thank Gloria Cain, Cynthia Marquardt, and Maria-Nera Davis for their administrative help and support.

Last but not the least; I want to extend my special thanks to my family members and friends who are always a constant and willing source of encouragement and inspiration.

ABSTRACT

Recent years have seen a rapid proliferation of information sources e.g., on the World-wide Web, in virtually every area of human endeavor. Such autonomous information sources are based on different ontologies, i.e., conceptualizations of the entities, properties, and relationships in the respective domains of discourse. However, practical applications (e.g., building predictive models from disparate data sources, assembling composite web services using components from multiple repositories) call for mechanisms that bridge the semantic gaps between disparate ontologies using mappings that express terms (concepts, properties, and relationships) in a target ontology in terms of those in one or more source ontologies. Such mappings may be established by domain experts or automatically using tools designed to discover such mappings from data. In either case, it is necessary to check whether the resulting mappings are consistent, and if necessary, make them consistent by eliminating a subset of the mappings. We consider the problem of identifying the largest (maximum) subset of mappings in the restricted, yet practically important setting of hierarchical ontologies. Specifically, we consider mappings that assert that a concept in one ontology is a subconcept, superconcept, or equivalent concept of a concept in another ontology. We model the problem of identifying the largest consistent subset of such mappings between hierarchical ontologies as the problem of identifying the minimum feedback arc set in a directed acyclic graph (DAG). Because identifying minimum feedback arc set is known to be NP-hard, it follows that identifying the maximum subset of consistent mappings between hierarchical ontologies is NP-hard. We then explore several polynomial time algorithms for finding suboptimal solutions including a heuristic algorithm for (weighted) minimum feedback arc set problem in DAGs. We compare the performance of the various algorithms on several synthetic as well as real-world ontologies and

mappings.

CHAPTER 1. INTRODUCTION

With the advent of the World Wide Web, there has been a pressing need for tools to handle the massive amount of distributed information as well as their integration. One of the most important fields of computing research that attempts to address this problem is the Semantic Web [Antoniou and Harmelen, 2008], which deals with how information is organized, understood and reasoned about by various parties and autonomous agents participating in the Web.

Of key interest to Semantic Web research is the semantic description of information. Due to the distributed nature of the Web, actors often want to share information among themselves. Naturally, this leads to the problem of information heterogeneity, which comes in the way of seamless sharing and reuse of information. In order for two actors to be able to share heterogeneous information in a common domain of interest, it is imperative that they come to a *common understanding* of the domain.

Information integration is a process of merging information from multiple sources. *Matching* – the process of identifying the correspondences between semantically related entities of the difference sources – is seen as a plausible solution for these kind of applications [Euzenat and Shvaiko, 2007]. Heterogeneity between multiple sources increase the difficulty of merging the information; and in distributed and open systems, such as semantic web, it is not possible to avoid heterogeneity.

1.1 Background

Before we discuss about the problem that we are addressing through this work, we will introduce a few basic concepts and terminologies in this section.

1.1.1 Ontology

Over the years, ontologies [Colomb, 2007] have emerged as the de facto choice for describing information semantics over the Web, using which one can attach semantics to his data. An *ontology* can be used within a domain to formally represent a set of *concepts* and the *relationships* between those concepts. It can be used both as a knowledge representation technique as well as to define the domain itself. Once the ontology is defined, it can be used to reason about the properties of that domain. In simple terms, an ontology is a "specification of a conceptualization" [Gruber, 1993]. Each ontology contains a set of primitive entities that can be used to model the domain that it represents. These entities primarily consist of the following:

- **Classes** or concepts represent the collections or types of objects or individuals. These are the main entities of an ontology.
- **Relationships** represent the set of relations or ways in which various classes are related to each other within the ontology
- **Individuals** or class instances represent a particular instance of a class in a domain.
- **Attributes** are used to capture the properties or characteristics of the classes

At its barest of essentials, an ontology can be viewed as describing concepts in a domain of interest and a classification of concepts in the domain. Hence, an ontology can also be defined as a set of concepts represented by *classes* and a classification of the concepts represented by *relationships*. Therefore, the *classes* and *relationships* capture the essential *structure* of the ontology, while attributes add supplementary information. There have been other extended representations of ontologies, that also consider axioms and other entities as part of an ontology specification. In this work, we restrict our treatment to *partial order ontologies*, i.e., ontologies specified by a set of concepts and an associated set of relationships that define a partial order over the concepts in the ontology [Bonatti et al., 2003].

For the rest of our discussion, we will assume concepts to represent named entities in the domain, and relationships to represent *binary* relations specifying one of *subclass*, *equivalence* or *superclass*, meaning that a concept is a *specialization* of, *equivalent* to or *generalization* of another concept respectively.

Various languages can be used to encode ontologies. OWL[1] is a W3C recommended language to represent ontologies. Although our work is not coupled with any ontology specification language, for the purpose of experimental evaluation of our work, we use OWL as the language of choice for ontology specification.

1.1.2 Combining Two Ontologies

Having seen that ontologies can be used effectively for sharing and reusing knowledge about information in the Web, the next question arises: how do we use the combined knowledge (in the form of ontology specification) of multiple entities? The task of *combining* ontologies of multiple actors in the Web, by *bridging* the *semantic gap* between their descriptions is crucial to realize the goal of Semantic Web — to enable seamless information integration over heterogeneous information sources.

There are several ways in which two ontologies can be combined. For example, *Ontology merging* is the process of generating a new ontology by combining two ontologies. The source ontologies may be overlapping with each other and after ontology merging they still remain unchanged. Apart from the information contained by both the source ontologies, the merged ontology contains additional information generated while merging. A slightly different process is *ontology integration*, which is the process of integrating one ontology within another ontology. After integration one of the ontology remains unchanged while the other ontology, after integration, contains the information of both of them. There are more ways in which two ontologies can be combined. Please refer [Euzenat and Shvaiko, 2007] for a discussion of various processes. The basic requirement for combining two ontologies is the knowledge about how to combine them. This process of finding relationships between classes of two ontologies is called *ontology matching*. These identified relationships are often called *alignment* or *mapping*.

[1]http://www.w3.org/TR/owl-features/

1.1.3 Problem

In this work, however, we focus our attention on the process of combining two ontologies using a set of already specified mappings between them. We do not try to identify how those mappings were identified.

Given any two ontologies and a user specified mapping set that specifies relationships between the classes of the two ontologies, we are interested in automatically identifying which of the mappings from the given mapping set can be used to combine the two ontologies such that the resulting ontology still remains *consistent* (does not contain contradictory knowledge such as, a concept is a subclass of itself).

In other words, given two consistent ontologies and a set of mappings between them, we are interested in methods to obtain a maximum subset of mappings such that when this mapping subset is used to combine the two ontologies, the resulting ontology still remains consistent.

1.2 Related Work

In the past, many (semi-) automated ways of generating these mappings have been proposed. They use various approaches to identify the mappings. For example, *schema-based* systems perform matching on the basis of schema level input. Another type of systems are *instance-based* and they rely on the instances or the data expressed by the ontology. Another type of systems combine both these approaches. A detailed study of these approaches is done in [Euzenat and Shvaiko, 2007].

Several tools have been developed using these techniques. A survey of these tools has been done by [Kalfoglou and Schorlemmer, 2003; Choi et al., 2006] and a detailed list of tools can be found at [Euzenat and Shvaiko, 2007, chap. 6]. Most of these approaches deal with automatically identifying the a set of mappings between two ontologies in order to facilitate sharing and reuse of knowledge. It must be noted that many approaches focus only on identifying a mapping set without worrying about the consistency of the merged ontology. In particular, [Falconer and Storey, 2007] deal with identifying mappings followed by user intervention where a user manually corrects the generated mappings. In fact, user involvement in ensuring consistency of mapping, specially in large ontologies, has been identified as one of the major challenges in ontology matching [Shvaiko and Euzenat, 2008].

To the best of our knowledge, there has been no work on identifying maximum mapping sets that ensure consistency. In this context, our work tries to address this problem of combining the ontologies by identifying the maximum subset of already identified mappings (for e.g. mappings identified by semi-automatic tools mentioned above) that ensures the consistency of the combined ontology. Our approach concerns with identifying *quantitatively more* mappings from the given set and is quality-agnostic, that is, we do not deal with the qualitative aspects of the mappings.

1.3 Contributions

Our contributions can be summarized as follows:

- We present the problem of identifying the maximum subset of a given mapping set that can be used to combine two consistent ontologies such that the resulting ontology also remains consistent.
- We prove that the problem is NP-hardusing a known hard problem in the graph domain.
- We discuss several polynomial time computable algorithms for finding suboptimal solutions of this problem. In particular we model the problem as a minimum feedback arc set problem in the graph domain and use a known heuristic to solve our problem.
- We compare the performance of our algorithms on several synthetic and real-world ontologies and mappings.

1.4 Organization

The remaining dissertation has been organized as follows:

- Chapter 2 formally describes the problem within our scope. We describe the terms ontology, mapping, and consistency as we refer them. Then we describe the problem statement and show that it is NP-hard.

- Chapter 3 is concerned with the various methods to solve the problem. We describe how we can represent our problem as an equivalent graph problem. Then we discuss a couple of simple heuristics and a graph-based heuristic that can all compute a solution for our problem in polynomial time.
- Chapter 4 presents our results. We compare the accuracy of our solution against the optimal solution. We also compare the various heuristics against each other.
- Chapter 5 summarizes our work.
- Appendix A summarizes the various notations that we use across all the chapters.
- Appendix B extends the scope of our problem by introducing some complex mappings. We also discuss a possible modification for our algorithm to solve the enhanced problem.

CHAPTER 2. PROBLEM DESCRIPTION

In this chapter we will formally describe the problem that we are trying to solve. Before introducing the problem, we will describe a few important terms and how we will refer them in this work.

2.1 Ontology

Definition 2.1 (Class). A *class* or a *concept* represents the collection or type of objects or individuals denoted by c.

Definition 2.2 (Relationship). Given a non-empty finite set of classes $\mathbb{C} = \{c_1, c_2, \dots\}$, we define a *relationship* r as a relation $c_i \mathcal{R} c_j$ between any two unique classes $c_i, c_j \in \mathbb{C}$ where $\mathcal{R}$ is either of the following two relations:

$\prec$ **Subclass relation.** $c_i \prec c_j$ represents that class c_i is a *subclass* of another class c_j where $\prec$ is a *strict partial order* relation, that is, all the following hold true:

- **Irreflexive.** $\neg (c_i \prec c_i)$
- **Asymmetric.** $\left(c_i \prec c_j\right) \Rightarrow \neg \left(c_j \prec c_i\right)$
- **Transitive.** $\left(c_i \prec c_j \wedge c_j \prec c_k\right) \Rightarrow (c_i \prec c_k)$

Sometimes, we also say that c_j is a super class of c_i and represent it as $c_j \succ c_i$. We note that

$$(c_i \prec c_j) \Leftrightarrow (c_j \succ c_i)$$

- $\equiv$ **Equivalence relation.** $c_i \equiv c_j$ represents that class c_i is *conceptually equivalent* to another class c_j where $\equiv$ is an equivalence[1] relation, that is, all the following hold true:

  - **Reflexive.** $(c_i \equiv c_i)$
  - **Symmetric.** $\left(c_i \equiv c_j\right) \Rightarrow \left(c_j \equiv c_i\right)$
  - **Transitive.** $\left(c_i \equiv c_j \wedge c_j \equiv c_k\right) \Rightarrow (c_i \equiv c_k)$

  Sometimes, we also say that c_i and c_j are *equivalent classes.*

Furthermore, the subclass relation and the equivalence relation are related as follows:

$$\left(c_i \prec c_j \wedge c_j \equiv c_k\right) \Rightarrow \left(c_i \prec c_k\right) \tag{2.1a}$$

$$\left(c_i \prec c_j \wedge c_i \equiv c_k\right) \Rightarrow \left(c_k \prec c_j\right) \tag{2.1b}$$

$$\left(c_i \prec c_j \wedge c_j \prec c_i\right) \not\Rightarrow \left(c_i \equiv c_j\right) \tag{2.1c}$$

$$\left(c_i \equiv c_j\right) \not\Rightarrow \left(c_i \prec c_j \wedge c_j \prec c_i\right) \tag{2.1d}$$

Definition 2.3 (Ontology)**.** An *ontology* is a two-tuple of a set of classes and a set of relationships between those classes denoted by $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ where $\mathsf{x} \in \mathbb{N}$ and,

$\mathbb{C}_{\mathsf{x}}$ is a non-empty finite set of classes in ontology $\mathcal{O}_{\mathsf{x}}$, that is, $\mathbb{C}_{\mathsf{x}} = \{c_1^{\mathsf{x}}, c_2^{\mathsf{x}}, \dots\}$

$\mathbb{R}_{\mathsf{x}}$ is a finite set of relationships in ontology $\mathcal{O}_{\mathsf{x}}$, that is, $\mathbb{R}_{\mathsf{x}} = \{r_1^{\mathsf{x}}, r_2^{\mathsf{x}}, \dots\}$

Example 2.1. For example, consider a very simple ontology that contains three classes a, b, and c, and two relationships specifying that a is a subclass of b and b is equivalent to c.

$$\mathcal{O}_1\colon \left\langle \{a, b, c\}\,,\ \{a \prec b, b \equiv c\} \right\rangle$$

Example 2.2. Now, consider the following example ontology extracted from the animalsA[2] ontology [Ehrig and Sure, 2005]:

$$\mathcal{O}_a\colon \left\langle \left\{ \begin{array}{l} Woman,\ Female,\ Person, \\ HumanBeing,\ Animal \end{array} \right\}, \left\{ \begin{array}{l} Woman \prec Female,\ Woman \prec Person, \\ Female \prec Animal,\ Person \prec Animal, \\ Person \equiv HumanBeing \end{array} \right\} \right\rangle$$

[1]We use the term equivalence in this context to represent the conceptual equivalence between two classes. It turns out that in our setting this binary relation is also mathematically an equivalence relation.

[2]Complete ontology at http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies/animalsA.owl.

We can observe that

$$\mathbb{C}_a = \{Woman,\ Female,\ Person,\ HumanBeing,\ Animal\}$$

that is, $\mathcal{O}_a$ contains five classes, viz, $Woman$, $Female$, $Person$, $HumanBeing$ and $Animal$ and

$$\mathbb{R}_a = \left\{ \begin{array}{l} Woman \prec Female,\ Woman \prec Person,\ Female \prec Animal, \\ Person \prec Animal,\ Person \equiv HumanBeing \end{array} \right\}$$

that is, $\mathcal{O}_a$ contains five relationships, viz. $Woman$ is a subclass of $Female$, $Woman$ is a subclass of $Person$, $Female$ is a subclass of $Animal$, $Person$ is a subclass of $Animal$, and $Person$ is a equivalent to $HumanBeing$.

Remark. We will conveniently use $\mathbb{O}$ to represent a set of all such ontologies, that is, $\mathbb{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots\}$

2.2 Ontology Graph

We can also represent each ontology as a graph that we refer to as an *ontology graph.* Given any ontology $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ the corresponding ontology graph is $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}\colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle$ where,

$\mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ is a non-empty finite set of vertices in the ontology graph of ontology $\mathcal{O}_{\mathsf{x}}$, that is, $\mathbb{V}_{\mathcal{O}_{\mathsf{x}}} = \{v_1^{\mathsf{x}}, v_2^{\mathsf{x}}, \dots\}$

v_i^{x} is the i^{th} vertex in the ontology graph of ontology $\mathcal{O}_{\mathsf{x}}$

Each vertex v_i^{x} represents a non-empty finite set of *equivalent classes* in $\mathbb{C}_{\mathsf{x}}$, that is, $v_i^{\mathsf{x}} = \{c_{i_1}^{\mathsf{x}}, c_{i_2}^{\mathsf{x}}, \dots\}$ where $c_{i_m}^{\mathsf{x}} \in \mathbb{C}_{\mathsf{x}}$ and such that exactly one of the following two conditions is true:

1. v_i^{x} is a singleton set and the only class contained in this set is not related to any other class in $\mathbb{C}_{\mathsf{x}}$ with the equivalence relation, that is, if $v_i^{\mathsf{x}} = \left\{ c_j^{\mathsf{x}} \right\}$ then,

$$\left(|v_i^{\mathsf{x}}| = 1\right) \wedge \left(\forall c_k^{\mathsf{x}} \in \mathbb{C}_{\mathsf{x}}\colon\ (c_j^{\mathsf{x}} \equiv c_k^{\mathsf{x}} \notin \mathbb{R}_{\mathsf{x}} \wedge c_k^{\mathsf{x}} \equiv c_j^{\mathsf{x}} \notin \mathbb{R}_{\mathsf{x}})\right)$$

2. Cardinality of v_i^{x} is more than 1 and each class in v_i^{x} is related to at least one other class in v_i^{x} with the equivalence relation, that is, if $v_i^{\mathsf{x}} = \left\{c_{i_1}^{\mathsf{x}}, c_{i_2}^{\mathsf{x}}, \dots\right\}$ then,

$$\left(|v_i^{\mathsf{x}}| > 1\right) \wedge \left(\forall c_{i_m}^{\mathsf{x}} \in v_i^{\mathsf{x}}\ \exists c_{i_n}^{\mathsf{x}} \in v_i^{\mathsf{x}} \colon \left(c_{i_m}^{\mathsf{x}} \equiv c_{i_n}^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}} \vee c_{i_n}^{\mathsf{x}} \equiv c_{i_m}^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}\right)\right)$$

$\mathbb{E}_{\mathcal{O}_{\mathsf{x}}}$ is a finite set of *directed ontology edges* in the ontology graph of ontology $\mathcal{O}_{\mathsf{x}}$, that is, $\mathbb{E}_{\mathcal{O}_{\mathsf{x}}} = \{e_1^{\mathsf{x}}, e_2^{\mathsf{x}}, \dots\}$

e_p^{x} is the p^{th} directed ontology edge in the ontology graph of ontology $\mathcal{O}_{\mathsf{x}}$

Each edge e_p^{x} represents a subclass relation such that there is a directed edge from the vertex containing the subclass to the vertex containing the super class, that is, if $\left(e_p^{\mathsf{x}} = v_i^{\mathsf{x}} \dashrightarrow v_j^{\mathsf{x}}\right)$ then,

$$\exists c_{i_m}^{\mathsf{x}} \in v_i^{\mathsf{x}}, \exists c_{j_n}^{\mathsf{x}} \in v_j^{\mathsf{x}} \colon c_{i_m}^{\mathsf{x}} \prec c_{j_n}^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}$$

As per this description of the ontology graph, we can easily identify the following property of the ontology graph:

Property 2.1. *For any ontology $\mathcal{O}_{\mathsf{x}} \colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$, corresponding ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}} \colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle$ contains at most $|\mathbb{C}_{\mathsf{x}}|$ vertices and at most $|\mathbb{R}_{\mathsf{x}}|$ edges, that is,*

$$|\mathbb{V}_{\mathcal{O}_{\mathsf{x}}}| \leq |\mathbb{C}_{\mathsf{x}}| \ \ \textit{and} \ \ |\mathbb{E}_{\mathcal{O}_{\mathsf{x}}}| \leq |\mathbb{R}_{\mathsf{x}}|$$

Example 2.3. For example, again consider ontology $\mathcal{O}_1 \colon \left\langle \{a, b, c\},\ \{a \prec b, b \equiv c\} \right\rangle$. The corresponding ontology graph $\mathcal{G}_{\mathcal{O}_1}$ is shown in Figure 2.1.

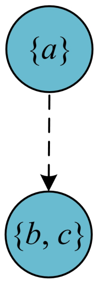


Figure 2.1: Ontology Graph $\mathcal{G}_{\mathcal{O}_1}$

2.2.1 Construction

Given some ontology $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$, the corresponding ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}\colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle$ can be generated using the following simple steps:

1. We will represent each set of equivalent classes as a single unique vertex. Therefore, we have:

$$\left(c_i^{\mathsf{x}} \equiv c_j^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}\right) \Leftrightarrow \exists v_l^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}\colon \left(\left(c_i^{\mathsf{x}}, c_j^{\mathsf{x}} \in v_l^{\mathsf{x}}\right) \wedge \left(\forall v_{l'}^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}\colon c_i^{\mathsf{x}}, c_j^{\mathsf{x}} \notin v_{l'}^{\mathsf{x}}\right)\right)$$
$$\left(c_i^{\mathsf{x}} \equiv c_j^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}} \wedge c_i^{\mathsf{x}} \equiv c_k^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}\right) \Leftrightarrow \exists v_l^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}\colon \left(\left(c_i^{\mathsf{x}}, c_j^{\mathsf{x}}, c_k^{\mathsf{x}} \in v_l^{\mathsf{x}}\right) \wedge \left(\forall v_{l'}^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}\colon c_i^{\mathsf{x}}, c_j^{\mathsf{x}}, c_k^{\mathsf{x}} \notin v_{l'}^{\mathsf{x}}\right)\right)$$

   Remark. We are only representing the classes and relationships between the classes. We are not concerned with the other entities of the ontology and they remain unchanged. When we use a single vertex to represent multiple classes, we do not worry about the attributes of those classes. The ontology still remains the same. We are combining the equivalent classes in a single vertex for simplicity.

2. For each class not covered in earlier step, we create a vertex each such that each vertex is a singleton set, containing only that class.

3. For each subclass relation ($\prec$), we add a directed edge ($\dashrightarrow$), from the subclass to the super class. Therefore, we have:

$$\left(c_{im}^{\mathsf{x}} \prec c_{jn}^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}\right) \Leftrightarrow \exists v_i^{\mathsf{x}}, v_j^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}\colon \left(c_{im}^{\mathsf{x}} \in v_i^{\mathsf{x}}\right) \wedge \left(c_{jn}^{\mathsf{x}} \in v_j^{\mathsf{x}}\right) \wedge \left(v_i^{\mathsf{x}} \dashrightarrow v_j^{\mathsf{x}} \in \mathbb{E}_{\mathcal{O}_{\mathsf{x}}}\right)$$

Algorithm 2.1 shows an algorithm to construct $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}\colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle$ from $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$. In the algorithm we look at each relationship at a time and based on the type of relation we create corresponding vertices and edge.

2.2.2 Complexity

In this construction, each class and each relationship is accessed once. Hence, it has a running time of $O\left(|\mathbb{C}| + |\mathbb{R}|\right)$.

Algorithm 2.1 Generating ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ from ontology $\mathcal{O}_{\mathsf{x}}$

```
Require: $\mathcal{O}_{\mathsf{x}} \colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$
 1: $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}} \colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle$, $\mathbb{V}_{\mathcal{O}_{\mathsf{x}}} := \emptyset$, $\mathbb{E}_{\mathcal{O}_{\mathsf{x}}} := \emptyset$
 2: $C := \mathbb{C}_{\mathsf{x}}$ // temporary //
    // $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{x}}$ results into a single vertex holding both of them //
 3: for all $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}$ do
 4:   if $\exists v_l^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ such that $c_i^{\mathsf{x}} \in v_l^{\mathsf{x}}$ then
 5:     $v_l^{\mathsf{x}} := v_l^{\mathsf{x}} \bigcup \left\{ c_j^{\mathsf{x}} \right\}$
 6:   else if $\exists v_l^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ such that $c_j^{\mathsf{x}} \in v_l^{\mathsf{x}}$ then
 7:     $v_l^{\mathsf{x}} := v_l^{\mathsf{x}} \bigcup \{ c_i^{\mathsf{x}} \}$
 8:   else
 9:     $v_l^{\mathsf{x}} := \left\{ c_i^{\mathsf{x}}, c_j^{\mathsf{x}} \right\}$, $\mathbb{V}_{\mathcal{O}_{\mathsf{x}}} := \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \bigcup \{ v_l^{\mathsf{x}} \}$
10:   end if
11:   $C := C \setminus \left\{ c_i^{\mathsf{x}}, c_j^{\mathsf{x}} \right\}$
12: end for
    // Create a vertex with singleton set for all the remaining classes //
13: for all $c_i^{\mathsf{x}} \in C$ do
14:   $v_l^{\mathsf{x}} := \{ c_i^{\mathsf{x}} \}$, $\mathbb{V}_{\mathcal{O}_{\mathsf{x}}} := \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \bigcup \{ v_l^{\mathsf{x}} \}$
15: end for
    // $c_i^{\mathsf{x}} \prec c_j^{\mathsf{x}}$ results into a directed edge //
16: for all $c_i^{\mathsf{x}} \prec c_j^{\mathsf{x}} \in \mathbb{R}_{\mathsf{x}}$ do
17:   Search $v_{i'}^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ such that $c_i^{\mathsf{x}} \in v_{i'}^{\mathsf{x}}$
18:   Search $v_{j'}^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ such that $c_j^{\mathsf{x}} \in v_{j'}^{\mathsf{x}}$
19:   $\mathbb{E}_{\mathcal{O}_{\mathsf{x}}} := \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \bigcup \left\{ v_{i'}^{\mathsf{x}} \dashrightarrow v_{j'}^{\mathsf{x}} \right\}$
20: end for
```

2.2.3 Function *vertex*

It is obvious from our construction that even though a vertex may contain multiple classes, each class is contained in one and only one vertex, that is,

$$\left(c_i^{\mathsf{x}} \in \mathbb{C}_{\mathsf{x}}\right) \Leftrightarrow \exists v_j^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \colon \left(\left(c_i^{\mathsf{x}} \in v_j^{\mathsf{x}}\right) \wedge \left(\forall v_{j'}^{\mathsf{x}} \in \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \colon c_i^{\mathsf{x}} \notin v_{j'}^{\mathsf{x}}\right) \right)$$

Based on this property, we now define a following useful function to obtain a vertex of any given class.

Definition 2.4 (vertex). For any ontology $\mathcal{O}_{\mathsf{x}}$ and the ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$, we define a function vertex: $\mathbb{C}_{\mathsf{x}} \xrightarrow{onto} \mathbb{V}_{\mathcal{O}_{\mathsf{x}}}$ as follows:

$$\text{vertex}\left(c_i^{\mathsf{x}}\right) = v_j^{\mathsf{x}} \text{ such that } c_i^{\mathsf{x}} \in v_j^{\mathsf{x}}$$

vertex is clearly a polynomial time computable function.

Simplifying Assumption. We can easily note here that we are actually merging all the equivalent classes in an ontology into a single vertex in the corresponding ontology graph. Since this merging is just the conceptual merging in the representation and not the actual merging of classes in the ontology, for simplicity, we will assume that the input ontologies do not contain any equivalence relationships. That is, all the relationships contained in the ontology are only the subclass relationships.

This implies that there is a unique vertex in the ontology graph such that each vertex is a singleton and each vertex corresponds to a particular unique class in the ontology. Hence, the Property 2.1 reduces to:

$$|\mathbb{V}_{\mathcal{O}_\mathsf{x}}| = |\mathbb{C}_\mathsf{x}| \text{ and } |\mathbb{E}_{\mathcal{O}_\mathsf{x}}| = |\mathbb{R}_\mathsf{x}|$$

Moreover, the definition of the function vertex also changes as follows:

$$\text{vertex}\colon \mathbb{C}_\mathsf{x} \xrightarrow{1:1} \mathbb{V}_{\mathcal{O}_\mathsf{x}} \text{ such that } \text{vertex}\left(c_i^\mathsf{x}\right) = v_i^\mathsf{x} = \left\{c_i^\mathsf{x}\right\}$$

2.3 Consistency in Ontology

Definition 2.5 (Inconsistent). An ontology $\mathcal{O}_\mathsf{x}\colon \langle \mathbb{C}_\mathsf{x}, \mathbb{R}_\mathsf{x} \rangle$ is said to be *inconsistent*, if the transitive closure of $\mathbb{R}_\mathsf{x}$ contains two or more relationships that contradict each other. Alternatively, an ontology $\mathcal{O}_\mathsf{x}\colon \langle \mathbb{C}_\mathsf{x}, \mathbb{R}_\mathsf{x} \rangle$ is said to be *inconsistent*, if the transitive closure of $\mathbb{R}_\mathsf{x}$ contains $c_i^\mathsf{x} \prec c_i^\mathsf{x}$.

Each subset of relationships that lead to some relationship $c_i^\mathsf{x} \prec c_i^\mathsf{x}$ in the transitive closure of the ontology is said to be a set of *conflicting relationships*. For any inconsistent ontology there may be one or more sets of conflicting relationships and each set may contain two or more relationships that conflict as a whole.

Definition 2.6 (Consistent). An ontology $\mathcal{O}_\mathsf{x}\colon \langle \mathbb{C}_\mathsf{x}, \mathbb{R}_\mathsf{x} \rangle$ is said to be *consistent* if it is not inconsistent.

Example 2.4. Consider some ontology $\mathcal{O}_2\colon \left\langle \{a,b,c\}\,,\ \{a \prec b, b \prec c\} \right\rangle$. We can observe that the closure is $\{a \prec b, b \prec c, a \prec c\}$. Since the closure does not contain any relationship $c_i^\times \prec c_i^\times$ for $i \in \{a,b,c\}$, hence $\mathcal{O}_2$ is consistent.

Example 2.5. Consider an ontology $\mathcal{O}_3\colon \left\langle \{a,b,c,d\}\,,\ \{a \prec b, c \prec a, d \prec b, d \prec c, a \prec d\} \right\rangle$. The corresponding ontology graph $\mathcal{G}_{\mathcal{O}_3}$ is shown in Figure 2.2.

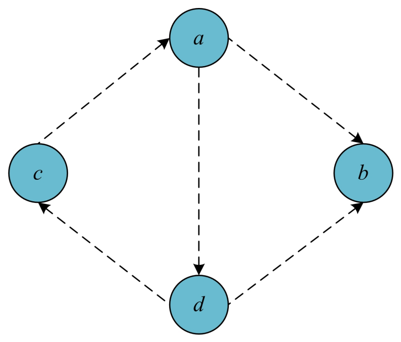


Figure 2.2: Ontology Graph $\mathcal{G}_{\mathcal{O}_3}$

We can observe that the closure is $\{a \prec b, c \prec a, d \prec b, d \prec c, a \prec d, a \prec a, c \prec c, d \prec d\}$. Since the closure contains relationships $a \prec a$, $c \prec c$, etc. hence, the ontology $\mathcal{O}_3$ is an inconsistent. Moreover, the only set of conflicting relationships is $\{c \prec a, d \prec c, a \prec d\}$.

Theorem 2.1. *An ontology $\mathcal{O}_\times$ is consistent if and only if its ontology graph $\mathcal{G}_{\mathcal{O}_\times}$ is a directed-acyclic graph (DAG).*

Proof. The proof is very simple since there is a one-to-one mapping between classes and relationships in ontology to vertices and edges in ontology graph.

$\Rightarrow$ First, we will try to prove that an ontology $\mathcal{O}_\times$ is consistent if its ontology graph $\mathcal{G}_{\mathcal{O}_\times}$ is a DAG. In order to do so, let us assume by contradiction that $\mathcal{O}_\times$ is a consistent ontology such that $\mathcal{G}_{\mathcal{O}_\times}$ is not a DAG.

1. Since $\mathcal{G}_{\mathcal{O}_\times}$ is not a DAG, there must be at least one cycle in $\mathcal{G}_{\mathcal{O}_\times}$. Let $\langle v_1^\times, v_2^\times, \ldots, v_n^\times \rangle$ denote any cycle such that $v_1^\times, v_2^\times, \ldots, v_n^\times \in \mathbb{V}_{\mathcal{O}_\times}$ and $v_1^\times \dashrightarrow v_2^\times, v_2^\times \dashrightarrow v_3^\times, \ldots, v_{n-1}^\times \dashrightarrow v_n^\times, v_n^\times \dashrightarrow v_1^\times \in \mathbb{E}_{\mathcal{O}_\times}$

2. $v_1^{\mathsf{x}} \dashrightarrow v_2^{\mathsf{x}} \Rightarrow c_1^{\mathsf{x}} \prec c_2^{\mathsf{x}}$,

   $v_2^{\mathsf{x}} \dashrightarrow v_3^{\mathsf{x}} \Rightarrow c_2^{\mathsf{x}} \prec c_3^{\mathsf{x}}$,

   $\dots$,

   $v_{n-1}^{\mathsf{x}} \dashrightarrow v_n^{\mathsf{x}} \Rightarrow c_{n-1}^{\mathsf{x}} \prec c_n^{\mathsf{x}}$

   $v_n^{\mathsf{x}} \dashrightarrow v_1^{\mathsf{x}} \Rightarrow c_n^{\mathsf{x}} \prec c_1^{\mathsf{x}}$

3. Hence, we have, $c_1^{\mathsf{x}} \prec c_2^{\mathsf{x}} \prec c_3^{\mathsf{x}} \prec \cdots \prec c_{n-1}^{\mathsf{x}} \prec c_n^{\mathsf{x}} \prec c_1^{\mathsf{x}}$ and by transitivity we have, $c_1^{\mathsf{x}} \prec c_1^{\mathsf{x}}$, $c_2^{\mathsf{x}} \prec c_2^{\mathsf{x}}$, and so on.

4. However, this relationship is not allowed in a consistent ontology. Hence, we get a contradiction. Thus, our assumption is invalid, that is, $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ must not contain any cycle. In other words, $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ must be a DAG.

$\Leftarrow$ Now, we will prove that if an ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ is DAG then the ontology $\mathcal{O}_{\mathsf{x}}$ is consistent. Let us assume by contradiction that $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ is a DAG such that $\mathcal{O}_{\mathsf{x}}$ is inconsistent ontology.

1. Since $\mathcal{O}_{\mathsf{x}}$ is inconsistent, there must be at least one set of conflicting relationships in $\mathcal{O}_{\mathsf{x}}$. Let, $\{c_1^{\mathsf{x}} \prec c_2^{\mathsf{x}},\ c_2^{\mathsf{x}} \prec c_3^{\mathsf{x}},\ \cdots,\ c_{n-1}^{\mathsf{x}} \prec c_n^{\mathsf{x}},\ c_n^{\mathsf{x}} \prec c_1^{\mathsf{x}}\}$ be the a set of conflicting relationships.

2. $c_1^{\mathsf{x}} \prec c_2^{\mathsf{x}} \Rightarrow v_1^{\mathsf{x}} \dashrightarrow v_2^{\mathsf{x}}$,

   $c_2^{\mathsf{x}} \prec c_3^{\mathsf{x}} \Rightarrow v_2^{\mathsf{x}} \dashrightarrow v_3^{\mathsf{x}}$,

   $\dots$,

   $c_{n-1}^{\mathsf{x}} \prec c_n^{\mathsf{x}} \Rightarrow v_{n-1}^{\mathsf{x}} \dashrightarrow v_n^{\mathsf{x}}$,

   $c_n^{\mathsf{x}} \prec c_1^{\mathsf{x}} \Rightarrow v_n^{\mathsf{x}} \dashrightarrow v_1^{\mathsf{x}}$

3. Hence, we have, $v_1^{\mathsf{x}} \dashrightarrow v_2^{\mathsf{x}} \dashrightarrow v_3^{\mathsf{x}} \dashrightarrow \cdots \dashrightarrow v_{n-1}^{\mathsf{x}} \dashrightarrow v_n^{\mathsf{x}} \dashrightarrow v_1^{\mathsf{x}}$ and thus, $\langle v_1^{\mathsf{x}}, v_2^{\mathsf{x}}, \dots, v_n^{\mathsf{x}} \rangle$ is a cycle.

4. We got a contradiction and therefore, $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ is not a DAG. Thus, our assumption that $\mathcal{O}_{\mathsf{x}}$ is inconsistent is invalid and this implies that $\mathcal{O}_{\mathsf{x}}$ is consistent.

Hence, we have proved that an ontology $\mathcal{O}_{\mathsf{x}}$ is consistent if and only if its ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}}$ is a directed-acyclic graph (DAG). $\square$

Definition 2.7 (consistent). We define a function consistent: $\mathbb{O} \longrightarrow \{\top, \bot\}$ as follows[3]:

$$\text{consistent}\left(\mathcal{O}_{\mathsf{x}}\right) = \begin{cases} \top & \text{if } \mathcal{G}_{\mathcal{O}_{\mathsf{x}}} \text{ is DAG} \\ \bot & \text{otherwise} \end{cases}$$

Therefore, the function consistent can be implemented using the topological ordering algorithm with a running time of $O\left(|\mathbb{V}| + |\mathbb{E}|\right)$ where $|\mathbb{V}|$ is the number of vertices and $|\mathbb{E}|$ is the number of edges in the graph [Kleinberg and Tardos, 2005]. Alternatively, we have a running time of $O\left(|\mathbb{C}| + |\mathbb{R}|\right)$ where $|\mathbb{C}|$ is the number of classes and $|\mathbb{R}|$ is the number of relationships in the ontology.

In Example 2.4 consistent $\left(\mathcal{O}_2\right) = \top$ and in Example 2.5 consistent $\left(\mathcal{O}_3\right) = \bot$.

2.4 Mapping

Definition 2.8 (Mapping Relationship). Given any two different ontologies $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}}\colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle$, a *mapping relationship* $r^{\mathsf{x:y}}$ is a relationship $c_i^{\mathsf{x}} \mathcal{R} c_j^{\mathsf{y}}$ between any two classes $c_i^{\mathsf{x}} \in \mathbb{C}_{\mathsf{x}}$, $c_j^{\mathsf{y}} \in \mathbb{C}_{\mathsf{y}}$ where $\mathcal{R}$ is either of the following[4]:

$\prec$ **Subclass relation.** $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}$ represents that class c_i^{x} is subclass of another class c_j^{y}

$\succ$ **Super class relation.** $c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}}$ represents that class c_i^{x} is super class of another class c_j^{y}

$\equiv$ **Equivalence relation.** $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}}$ represents that class c_i^{x} is equivalent to another class c_j^{y}

We note that the properties of the relations are as defined in Section 2.1.

Definition 2.9 (Mapping Set). Given any two ontologies $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}}\colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle$, a *mapping set* $\mathbb{M}_{\mathsf{x:y}}$ is a set of mapping relationships, that is, $\mathbb{M}_{\mathsf{x:y}} = \left\{r_1^{\mathsf{x:y}}, r_2^{\mathsf{x:y}}, \dots\right\}$.

Example 2.6. Consider the following two ontologies:

$$\mathcal{O}_4\colon \Big\langle \{a, b, c\},\ \{c \prec b, b \prec a\} \Big\rangle \text{ and } \mathcal{O}_5\colon \Big\langle \{x, y, z\},\ \{z \prec y, y \prec x\} \Big\rangle$$

It is possible to have several mapping sets between these two ontologies. One possible mapping set is $\mathbb{M}_{4:5} = \{a \succ x, b \succ y, c \equiv z\}$.

[3] $\top$ denotes *true* and $\bot$ denotes *false*

[4] It is possible to define a mapping relationship as a relation between two sets of classes on either ontologies. We discuss those relationships and the changes required to be made to our solution in Appendix B.

2.5 Problem Statement

Generally, there are several ways to combine two ontologies with the help of mappings, resulting in a new ontology. For an overview of such methods please refer [Euzenat and Shvaiko, 2007].

We can combine two ontologies, to generate a new combined ontology, using a mapping set given between them. This combined ontology contains all the classes and the relationships that were contained in the given ontologies. In addition, this ontology also contains all the relationships specified by the mapping set. In literature this is also known as *ontology merging*.

Given any two ontologies $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}}\colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle$, and some mapping set $\mathbb{M}_{\mathsf{x:y}}$ we can generate a *merged ontology* $\mathcal{O}_{\mathsf{z}}\colon \langle \mathbb{C}_{\mathsf{z}},\ \mathbb{R}_{\mathsf{z}} \rangle$ by adding the relationships specified in the mapping set. We will often use $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}\colon \langle \mathbb{C}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{R}_{\mathbb{M}_{\mathsf{x:y}}} \rangle$ to represent a merged ontology that is generated by combining the ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ using mapping set $\mathbb{M}_{\mathsf{x:y}}$. Unless specified otherwise, we will also use the following additional notations:

$|\mathbb{C}| = |\mathbb{C}_{\mathsf{x}}| + |\mathbb{C}_{\mathsf{y}}|$ to represent the total number of classes in the merged ontology

$|\mathbb{R}| = |\mathbb{R}_{\mathsf{x}}| + |\mathbb{R}_{\mathsf{y}}|$ to represent the total number of relationships in the original ontologies (please note that this does not include the mapping relationships)

$|\mathbb{M}| = |\mathbb{M}_{\mathsf{x:y}}|$ represents the number of mapping relationships

2.5.1 Maximum Consistent Mapping Subset

Definition 2.10 (Consistent Mapping Subset)**.** Given any two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, and some mapping set $\mathbb{M}_{\mathsf{x:y}}$, a subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ is said to be a *consistent mapping subset* if $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ is a consistent[5] ontology, that is, consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$.

Example 2.7. Again consider the following two consistent ontologies

$$\mathcal{O}_4\colon \Big\langle \{a, b, c\}\,,\ \{c \prec b, b \prec a\} \Big\rangle \text{ and } \mathcal{O}_5\colon \Big\langle \{x, y, z\}\,,\ \{z \prec y, y \prec x\} \Big\rangle$$

[5] We will show in Section 3.3 how we can use the same consistent function to compute the consistency of a merged ontology.

and a mapping set $\mathbb{M}_{4:5} = \{a \succ x, b \succ y, c \equiv z\}$. With the help of transitive closure, it is easy to verify that all the following sets are consistent mapping subsets of $\mathbb{M}_{4:5}$:

- $\{\}$
- $\{a \succ x\}$
- $\{b \succ y\}$
- $\{c \equiv z\}$
- $\{a \succ x, b \succ y\}$

Definition 2.11 (Maximal Consistent Mapping Subset)**.** Given any two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, and some mapping set $\mathbb{M}_{\mathsf{x:y}}$, a consistent mapping subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ is said to be a *maximal consistent mapping subset* if adding one more mapping to $\mathbb{M}'_{\mathsf{x:y}}$ will make $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ inconsistent. That is, $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ is a maximal consistent mapping subset if $\left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top\right)$ and

$$\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} : \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right) \text{ where } \mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \{r_p^{\mathsf{x:y}}\}\right)$$

Example 2.8. In Example 2.7 the following two consistent mapping subsets are also maximal consistent mapping subsets:

- $\{c \equiv z\}$
- $\{a \succ x, b \succ y\}$

As exemplified in Example 2.8, for any given pair of consistent ontologies and a mapping set between those two ontologies, there can be multiple maximal consistent mapping subsets. Moreover, either of these subsets may be of interest to the user and it is difficult to identify the particular subset that the user may be interested in. One possible way in which the user may specify preference is by specifying positive integral weight for each mapping such that a mapping relationship with higher weight is preferred over the mapping relationship with lower weight.

Definition 2.12 (Weighted Mapping Set)**.** Given ontologies $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}}\rangle$ and $\mathcal{O}_{\mathsf{y}}\colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}}\rangle$, a *weighted mapping set* $\mathbb{M}_{\mathsf{x:y}}$ is a set of mapping relationships, that is, $\mathbb{M}_{\mathsf{x:y}} = \left\{r_1^{\mathsf{x:y}}, r_2^{\mathsf{x:y}}, \dots\right\}$ along with a weight function $\omega\colon \mathbb{M}_{\mathsf{x:y}} \longrightarrow \mathbb{N}$ where $\mathbb{N}$ is the set of positive integers defined as follows:

$$\omega\left(r_p^{\mathsf{x:y}}\right) = n \text{ where } n \in \mathbb{N}$$

Note. We will often use the following shorthand notation:

$$\omega^{\Sigma}\left(\mathbb{M}_{\mathsf{x:y}}\right) = \sum_{\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}}} \omega\left(r_p^{\mathsf{x:y}}\right)$$

Example 2.9. Again, consider the following two ontologies:

$$\mathcal{O}_4\colon \Big\langle \{a, b, c\}\,,\ \{c \prec b, b \prec a\} \Big\rangle \text{ and } \mathcal{O}_5\colon \Big\langle \{x, y, z\}\,,\ \{z \prec y, y \prec x\} \Big\rangle$$

One possible weighted mapping set is $\mathbb{M}_{4:5} = \{a \succ x, b \succ y, c \equiv z\}$, and

- $\omega\left(a \succ x\right) = 4$
- $\omega\left(b \succ y\right) = 2$
- $\omega\left(c \equiv z\right) = 3$

Note. Here onwards, we assume that all our mapping set are weighted mapping set, that is, they include a ω function. If the ω is not specified, then we assume the weight for each mapping relationship to be a constant unit weight.

Definition 2.13 (Maximum Consistent Mapping Subset)**.** Given any two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, and some weighted mapping set $\mathbb{M}_{\mathsf{x:y}}$, a maximal consistent mapping subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ is said to be a *maximum consistent mapping subset* if sum of weights of the mappings is maximized. That is, $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ is a maximum consistent mapping subset if $\mathbb{M}'_{\mathsf{x:y}}$ is a maximal consistent mapping subset and

$$\forall \mathbb{M}''_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}\colon \omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq \omega^{\Sigma}\left(\mathbb{M}''_{\mathsf{x:y}}\right)$$

where $\mathbb{M}''_{\mathsf{x:y}}$ is a maximal consistent mapping subset.

Again, for any given pair of consistent ontologies and a weighted mapping set between those two ontologies, there can be multiple maximum consistent mapping subsets. Moreover, if the weight of each mapping relationship is same, then it turns out that the maximum consistent mapping subset is a maximal consistent mapping subset with the highest cardinality.

2.5.2 Optimization Version

Given two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, and some weighted mapping set $\mathbb{M}_{\mathsf{x:y}}$, identify a maximum consistent mapping subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$. We denote this problem as McM.

McM: Given consistent $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and some $\mathbb{M}_{\mathsf{x:y}}$, find a subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} \colon \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$
3. $\forall \mathbb{M}''_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}} \colon \omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq \omega^{\Sigma}\left(\mathbb{M}''_{\mathsf{x:y}}\right)$ where $\mathbb{M}''_{\mathsf{x:y}}$ is a maximal consistent mapping subset

Remark. Later, we will show how this problem can be modeled as a minimum feedback arc set problem in a weighted directed graph.

2.5.3 Decision Version

Given two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, some weighted mapping set $\mathbb{M}_{\mathsf{x:y}}$, and a number $k \in \mathbb{N}$, is there a maximal consistent mapping subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ of weight at least k? We denote this problem as McM_d.

McM_d: Given consistent $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$, some $\mathbb{M}_{\mathsf{x:y}}$, and some $k \in \mathbb{N}$, is there $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} \colon \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$
3. $\omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq k$

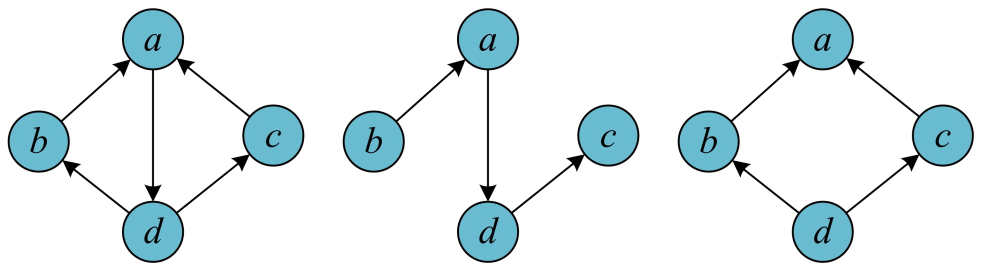
2.6 Complexity

In this section we will show that the decision problem McM$_d$ is NP-complete and the corresponding optimization problem is McM is NP-hard. We will show these results by reducing a known NP-complete problem, namely, Minimum Feedback Arc Set in Bipartite Tournament (MFASBT) [Guo et al., 2007] to our problem. We will now describe the feedback arc set.

2.6.1 Feedback Arc Set (FAS)

Given a directed graph $\mathcal{G}\colon \langle \mathbb{V},\ \mathbb{A}\rangle$ where $\mathbb{V}$ is the set of vertices and $\mathbb{A}$ is the set of directed edges (arcs), *feedback arc set* of $\mathcal{G}$ is a subset of edges, $\mathbb{A}' \subseteq \mathbb{A}$ such that $\mathcal{G}'\colon \langle \mathbb{V},\ \mathbb{A} \setminus \mathbb{A}'\rangle$ is acyclic. We will often use the shorthand $\mathcal{G} \setminus \mathbb{A}'$ to mean $\mathcal{G}'$. In simple words, feedback arc set of a directed graph is the set of those edges, which when removed from the directed graph would leave the graph cycle-free, that is, a directed acyclic graph (DAG).

Example 2.10. Figure 2.3 shows an example. Figure 2.3(b) and Figure 2.3(c) show two different subgraphs (DAGs) after removal of different feedback arc sets from the digraph shown Figure 2.3(a).



(a) Graph with cycles (b) Graph without a FAS (c) Graph without MFAS

Figure 2.3: Example for Feedback Arc Set

2.6.2 Minimum Feedback Arc Set (MFAS)

Minimum feedback arc set (MFAS) problem for a directed graph is the problem of finding a minimum set of edges (set with minimum cardinality) to be removed in order to break all the cycles in the graph. Karp showed that this problem is NP-hard [Karp, 1972; Garey and

Johnson, 1979]. A few years back it was shown that MFAS is NP-hard even for tournament graphs [Alon, 2006; Charbit et al., 2007]. Soon after that it was shown that MFAS is NP-complete even for bipartite tournament graphs [Guo et al., 2007]. We use the problem of finding minimum feedback arc set in bipartite tournament (MFASBT) to prove the hardness of McM.

Example 2.11. Figure 2.3 shows an example. Figure 2.3(c) is a subgraph after removing the minimum feedback arc set from Figure 2.3(a).

Definition 2.14 (Bipartite Tournament)**.** A *tournament graph* is a directed graph where there is exactly one directed edge between each pair of vertices. In other words, it is some directed orientation of a complete undirected graph. A *bipartite tournament graph* is a directed orientation of a complete bipartite undirected graph. We will denote a bipartite tournament graph as $\mathcal{G}\colon \langle \mathbb{X}, \mathbb{A}, \mathbb{Y}\rangle$ where $\mathbb{X}$ and $\mathbb{Y}$ are the bipartite sets of vertices and $\mathbb{A}$ is set of directed edge between them.

Example 2.12. Figure 2.4 shows an example of a bipartite tournament graph with the bipartite vertex sets being $\{x_1, x_2\}$ and $\{y_1, y_2\}$.

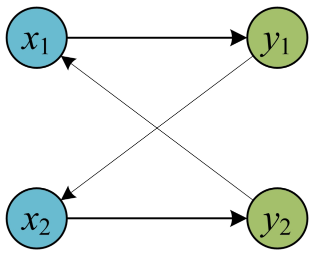


Figure 2.4: Example for Bipartite Tournament Graph

2.6.3 Problem Complexity

Now, we are ready to prove the hardness of the decision problem McM_d.

Theorem 2.2. McM_d *is NP-complete where*

McM_d: *Given consistent* $\mathcal{O}_{\mathsf{x}}$ *and* $\mathcal{O}_{\mathsf{y}}$, *some* $\mathbb{M}_{\mathsf{x:y}}$, *and some* $k \in \mathbb{N}$, *is there* $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ *such that all the following are true:*

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$

2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} \colon \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ *where* $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$

3. $\omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq k$

Proof. The proof is as follows:

- **Proof for** $\text{McM}_d \in NP$**.** We prove this by showing a polynomial time computable certifier below.

  - **Certificate.** A certificate is any given subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$.
  - **Certifier.** The certifier returns *yes* if all the following conditions are true:

    1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
    2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} \colon \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$
    3. $\omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq k$

    Otherwise, it returns *no*.

- **Known NP-complete problem.** Now, we need to select a known NP-complete problem and then we will reduce it to our problem in polynomial time. We pick the problem of identifying the minimum feedback arc set in a bipartite tournament [Guo et al., 2007] specified as follows:

  MFASBT: Given a bipartite tournament graph $\mathcal{G} \colon \langle \mathbb{X}, \mathbb{A}, \mathbb{Y} \rangle$ where $\mathbb{X}$ and $\mathbb{Y}$ are bipartite sets of vertices and $\mathbb{A}$ is set of directed arcs between the bipartite and a number $k \in \mathbb{N}$, is there a feedback arc set of size at most k, that is, is there a subset $\mathbb{A}' \subseteq \mathbb{A}$ such that $|\mathbb{A}'| \leq k$ and $\mathcal{G} \setminus \mathbb{A}'$ is acyclic?

- **Proof for** $\text{MFASBT} \leq_P \text{McM}_d$**.** Now, we need to show that MFASBT is polynomial time reducible to McM_d.

  1. **Construction.** First we need to show how we can transform any instance of MFASBT into an instance of McM_d. Let us assume $\langle \mathcal{G} \colon \langle \mathbb{X}, \mathbb{A}, \mathbb{Y} \rangle, k \rangle$ to be an in-

stance of MFASBT. We will transform it to an instance of McM$_d$, $\langle \mathcal{O}_{\mathsf{x}}, \mathcal{O}_{\mathsf{y}}, \mathbb{M}_{\mathsf{x:y}}, k \rangle$, where k remains unchanged.

Ontologies. We can generate ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ for the bipartite $\mathbb{X}$ and $\mathbb{Y}$ respectively such that $\mathcal{O}_{\mathsf{x}}$ contains $|\mathbb{X}| + 1$ classes and $\mathcal{O}_{\mathsf{y}}$ contains $|\mathbb{Y}| + 1$ classes, noting that each ontology contains a class more than the number of vertices. In each ontology, the extra class is made the super class of all the other classes. These are the only relationships in those ontologies.

Mapping. Now, for each edge $x_i \longrightarrow y_j \in \mathbb{A}$ and $y_j \longrightarrow x_i \in \mathbb{A}$ where $x_i \in \mathbb{X}$, $y_j \in \mathbb{Y}$, we generate a mapping relationship $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ and $c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ respectively and assign it a unit weight.

This completes our construction. Algorithm 2.2 shows this polynomial time computable transformation.

Example 2.13. An instance of MFASBT as shown in Figure 2.4 (with some k) will get transformed into following instance of McM$_d$:

$$\mathcal{O}_{\mathsf{x}}\colon \left\langle \left\{c_0^{\mathsf{x}}, c_1^{\mathsf{x}}, c_2^{\mathsf{x}}\right\}, \left\{c_1^{\mathsf{x}} \prec c_0^{\mathsf{x}}, c_2^{\mathsf{x}} \prec c_0^{\mathsf{x}}\right\}\right\rangle,\ \mathcal{O}_{\mathsf{y}}\colon \left\langle \left\{c_0^{\mathsf{y}}, c_1^{\mathsf{y}}, c_2^{\mathsf{y}}\right\}, \left\{c_1^{\mathsf{y}} \prec c_0^{\mathsf{y}}, c_2^{\mathsf{y}} \prec c_0^{\mathsf{y}}\right\}\right\rangle,$$

$$\mathbb{M}_{\mathsf{x:y}} = \left\{c_1^{\mathsf{x}} \prec c_1^{\mathsf{y}}, c_1^{\mathsf{x}} \succ c_2^{\mathsf{y}}, c_2^{\mathsf{x}} \prec c_2^{\mathsf{y}}, c_2^{\mathsf{x}} \prec c_1^{\mathsf{y}}\right\},$$

$$\omega\left(c_1^{\mathsf{x}} \prec c_1^{\mathsf{y}}\right) = \omega\left(c_1^{\mathsf{x}} \succ c_2^{\mathsf{y}}\right) = \omega\left(c_2^{\mathsf{x}} \prec c_2^{\mathsf{y}}\right) = \omega\left(c_2^{\mathsf{x}} \prec c_1^{\mathsf{y}}\right) = 1$$

Now, we must note the following properties of this transformation:

(a) Each simple cycle[6] in $\mathcal{G}$ contains 4 or more even number of vertices and each adjacent vertex is from different bipartite

(b) For any arbitrary simple cycle

$$x_i \longrightarrow y_j \longrightarrow x_k \longrightarrow \cdots \longrightarrow y_l \longrightarrow x_i$$

[6]A simple cycle is a cycle in which each vertex and each edge participates only once.

Algorithm 2.2 Algorithm to transform any instance of MFASBT into an instance of McM$_d$

```
Require: $\mathcal{G}$: $\langle \mathbb{X}, \mathbb{A}, \mathbb{Y} \rangle$, $k$
 1: $\mathcal{O}_{\mathsf{x}}$: $\langle \mathbb{C}_{\mathsf{x}}, \mathbb{R}_{\mathsf{x}} \rangle$, $\mathbb{C}_{\mathsf{x}} := \{c_0^{\mathsf{x}}\}$, $\mathbb{R}_{\mathsf{x}} := \emptyset$
 2: $\mathcal{O}_{\mathsf{y}}$: $\langle \mathbb{C}_{\mathsf{y}}, \mathbb{R}_{\mathsf{y}} \rangle$, $\mathbb{C}_{\mathsf{y}} := \left\{c_0^{\mathsf{y}}\right\}$, $\mathbb{R}_{\mathsf{y}} := \emptyset$
 3: $\mathbb{M}_{\mathsf{x:y}} := \emptyset$
 4: $k$ remains same in the transformed instance
 5: for $i = 1$ to $|\mathbb{X}|$ do
 6:     $\mathbb{C}_{\mathsf{x}} := \mathbb{C}_{\mathsf{x}} \bigcup \{c_i^{\mathsf{x}}\}$
 7:     $\mathbb{R}_{\mathsf{x}} := \mathbb{R}_{\mathsf{x}} \bigcup \{c_i^{\mathsf{x}} \prec c_0^{\mathsf{x}}\}$
 8: end for
 9: for $j = 1$ to $|\mathbb{Y}|$ do
10:     $\mathbb{C}_{\mathsf{y}} := \mathbb{C}_{\mathsf{y}} \bigcup \left\{c_j^{\mathsf{y}}\right\}$
11:     $\mathbb{R}_{\mathsf{y}} := \mathbb{R}_{\mathsf{y}} \bigcup \left\{c_j^{\mathsf{y}} \prec c_0^{\mathsf{y}}\right\}$
12: end for
13: for all arc $x_i \longrightarrow y_j \in \mathbb{A}$ where $x_i \in \mathbb{X}$ and $y_j \in \mathbb{Y}$ do
14:     $\mathbb{M}_{\mathsf{x:y}} := \mathbb{M}_{\mathsf{x:y}} \bigcup \left\{c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}\right\}$
15:     $\omega\left(c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}\right) := 1$
16: end for
17: for all arc $y_j \longrightarrow x_i \in \mathbb{A}$ where $x_i \in \mathbb{X}$ and $y_j \in \mathbb{Y}$ do
18:     $\mathbb{M}_{\mathsf{x:y}} := \mathbb{M}_{\mathsf{x:y}} \bigcup \left\{c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}}\right\}$
19:     $\omega\left(c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}}\right) := 1$
20: end for
```

we will have following mapping relationships:

$$c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}},\ c_k^{\mathsf{x}} \succ c_j^{\mathsf{y}},\ c_k^{\mathsf{x}} \prec \cdots,\ \ldots,\ \cdots \prec c_l^{\mathsf{y}},\ c_i^{\mathsf{x}} \succ c_l^{\mathsf{y}} \tag{2.2}$$

$$\Rightarrow c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}} \prec c_k^{\mathsf{x}} \prec \cdots \prec c_l^{\mathsf{y}} \prec c_i^{\mathsf{x}} \tag{2.3}$$

$$\Rightarrow c_i^{\mathsf{x}} \prec c_i^{\mathsf{x}},\ c_j^{\mathsf{y}} \prec c_j^{\mathsf{y}},\ c_k^{\mathsf{x}} \prec c_k^{\mathsf{x}},\ \ldots,\ c_l^{\mathsf{y}} \prec c_l^{\mathsf{y}} \tag{2.4}$$

Now, this closure contains the relationships $c_i^{\mathsf{x}} \prec c_i^{\mathsf{x}}$, etc. hence, $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$ is inconsistent. Moreover, $\left\{c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}},\ c_k^{\mathsf{x}} \succ c_j^{\mathsf{y}},\ c_k^{\mathsf{x}} \prec \cdots,\ \ldots,\ \cdots \prec c_l^{\mathsf{y}},\ c_i^{\mathsf{x}} \succ c_l^{\mathsf{y}}\right\}$ is a set of conflicting relationships. Further, it can be noted that if this cycle was absent, Equation (2.3) will also be absent in the closure, hence, there will be no relationships $c_i^{\mathsf{x}} \prec c_i^{\mathsf{x}}$. That is, any simple cycle in $\mathcal{G}$ implies inconsistency of $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$.

(c) Conversely, any inconsistency in $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ implies a cycle in $\mathcal{G}$. This is because any set of conflicting relationships in $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ can not involve the additional classes c_0^{x} or c_0^{y} as no other class is their super class. That is, they can not participate in a sequence of relationships as in Equation (2.3). Therefore, there is a set of edges in $\mathcal{G}$ corresponding to the conflicting relationships, which would be a cycle as per our construction. Hence, any inconsistency in $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ implies a cycle in $\mathcal{G}$.

2. *yes* **instance of** MFASBT**.** Given a *yes* instance of MFASBT, $\mathbb{A}' \subseteq \mathbb{A}$ will lead to a *yes* instance of McM$_d$. This is because, $\mathcal{G} \setminus \mathbb{A}'$ does not contain any cycle, which means the transformed instance of $\mathcal{G} \setminus \mathbb{A}'$, $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ is consistent, as discussed above.

3. *no* **instance of** MFASBT**.** Given a *no* instance of MFASBT, $\mathbb{A}' \subseteq \mathbb{A}$ will lead to a *no* instance of McM$_d$. This is because, $\mathcal{G} \setminus \mathbb{A}'$ contains some cycle, which means the transformed instance of $\mathcal{G} \setminus \mathbb{A}'$, $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ is inconsistent, as discussed above.

□

Theorem 2.3. McM *is NP-hard where*

McM: *Given consistent* $\mathcal{O}_{\mathsf{x}}$ *and* $\mathcal{O}_{\mathsf{y}}$ *and some* $\mathbb{M}_{\mathsf{x:y}}$*, find a subset* $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ *such that all the following are true:*

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$

2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}} : \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ *where* $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$

3. $\forall \mathbb{M}''_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}} : \omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq \omega^{\Sigma}\left(\mathbb{M}''_{\mathsf{x:y}}\right)$ *where* $\mathbb{M}''_{\mathsf{x:y}}$ *is a maximal consistent mapping subset*

Proof. The proof follows from Theorem 2.2. □

CHAPTER 3. METHODS AND PROCEDURES

In Section 2.6.3 we showed that McM is NP-hard and hence, it is not possible to have a polynomial time computable function to find the optimal solution. Hence, in this chapter we try to identify some algorithms that can compute near-good sub-optimal solution in polynomial time. Before we do that, we will try to take a look at a possible brute force solution for the problem.

3.1 Brute Force Approach

Let us recall that the problem that we are trying to solve is,
McM: Given consistent $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and some $\mathbb{M}_{\mathsf{x:y}}$, find a subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}}\colon \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \left\{r_p^{\mathsf{x:y}}\right\}\right)$
3. $\forall \mathbb{M}''_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}\colon \omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) \geq \omega^{\Sigma}\left(\mathbb{M}''_{\mathsf{x:y}}\right)$ where $\mathbb{M}''_{\mathsf{x:y}}$ is a maximal consistent mapping subset

3.1.1 Approach

A brute force approach is to first try to add all the mappings and test if the combined ontology is consistent or not. If the combined ontology is not consistent then remove one mapping at a time and test if the combined ontology is consistent or not. This way we will have $|\mathbb{M}_{\mathsf{x:y}}|$ different subsets each of size $|\mathbb{M}_{\mathsf{x:y}}| - 1$ and among all the consistent mapping subsets of that cardinality we pick the one that has highest combined weight. If none of the combined ontology is consistent, then remove two mappings at a time and test if the combined ontology is consistent or not. This way we will have $|\mathbb{M}_{\mathsf{x:y}}|\left(|\mathbb{M}_{\mathsf{x:y}}|-1\right)/2$ different subsets each of size $|\mathbb{M}_{\mathsf{x:y}}|-2$ and again, among all the consistent mapping subsets of that cardinality we pick the one that has highest combined weight. We keep on doing this until a consistent merged ontology is found. The subset of mappings with the highest weight in that iteration is a maximum consistent mapping subset.

3.1.2 Algorithm

Algorithm 3.1 shows a simple implementation of the above approach.

Algorithm 3.1 Brute Force Approach to solve McM

```
Require: $\mathcal{O}_{\mathsf{x}}$, $\mathcal{O}_{\mathsf{y}}$, $\mathbb{M}_{\mathsf{x:y}}$
 1: $result := \emptyset$
 2: for $n = |\mathbb{M}_{\mathsf{x:y}}|$ to 1 do
 3:   for all $\mathbb{M}'_{\mathsf{x:y}} \in \wp\left(\mathbb{M}_{\mathsf{x:y}}\right)$ where $\left|\mathbb{M}'_{\mathsf{x:y}}\right| = n$ do
 4:     if consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$ and $\omega^{\Sigma}\left(\mathbb{M}'_{\mathsf{x:y}}\right) > \omega^{\Sigma}\left(result\right)$ then
 5:       $result := \mathbb{M}'_{\mathsf{x:y}}$
 6:     end if
 7:   end for
 8:   if $result \neq \emptyset$ then
 9:     return $result$
10:   end if
11: end for
12: return $\emptyset$
```

3.1.3 Correctness

Essentially in this approach, we are generating the power set $\wp\left(\mathbb{M}_{\mathsf{x:y}}\right)$ of the mapping set and then we are using one subset $\mathbb{M}'_{\mathsf{x:y}} \in \wp\left(\mathbb{M}_{\mathsf{x:y}}\right)$ at a time, in the decreasing order of cardinality. Since we are checking all the possible subsets of a larger cardinality before checking any subset of a smaller cardinality, hence, the subset that we get is a maximum subset.

3.1.4 Complexity

We know that $|\wp(\mathbb{M}_{\mathsf{x:y}})| = 2^{|\mathbb{M}_{\mathsf{x:y}}|}$. Now, even if there is some very efficient way to generate power sets, we still need to verify an exponential number of subsets. Moreover, for each check, we need to generate the combined ontology and then verify its consistency. Hence, this is clearly not a good solution for large value of $|\mathbb{M}_{\mathsf{x:y}}|$.

3.1.5 Discussion

This approach in effect will exhaust the complete search space. Hence, the obvious problem with it is that it does not perform well when the number of mappings is large. Therefore, we shift our focus to algorithms which may compute sub-optimal result, however, they are computable much efficiently.

3.2 Approximate and Heuristic Approaches

Now, we will try to find a sub-optimal solution for our problem such that it is computable in polynomial time. That is, now we will try to find a maximal consistent mapping subset. The higher the weight of the computed subset the closer we will be to the optimal solution and the better our algorithm. Let us specify the simplified problem below:

McM$_m$: Given consistent $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and some $\mathbb{M}_{\mathsf{x:y}}$, find a subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}}$: $\left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \{r_p^{\mathsf{x:y}}\}\right)$

There are few simple and straight-forward approaches to compute a maximal consistent mapping subset that we discuss later in Section 3.4 and Section 3.5. However, before doing that we will show how we can compute the function consistent.

3.3 Mapping Graph

Earlier, in proof of Theorem 2.2, we used a graph problem to prove that our problem is also NP-hard. Hence, while trying to solve the problem, we also looked at the heuristics that may be used to solve the similar graph problems. Demetrescu and Finocchi have identified a combinatorial algorithm for finding feedback arc set in a weighted directed graph [Demetrescu and Finocchi, 2003]. Given a weighted directed graph their algorithm can compute a minimal weight feedback arc set, such that the approximation ratio is bounded by the length of the longest simple cycle. In order to use their algorithm we need to identify a way to convert our problem into a feedback arc set in weighted directed graph problem. We do so, by introducing a mapping graph – a graph that we will use to represent the problem such that we can clearly distinguish between the ontology relationships and mapping relationships.

Given any two ontologies $\mathcal{O}_{\mathsf{x}}\colon \langle \mathbb{C}_{\mathsf{x}}, \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}}\colon \langle \mathbb{C}_{\mathsf{y}}, \mathbb{R}_{\mathsf{y}} \rangle$ and a mapping set $\mathbb{M}_{\mathsf{x:y}}$, the corresponding *mapping graph* is $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ where,

$\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$ is a finite non-empty set of vertices such that

$$\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} = \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \bigcup \mathbb{V}_{\mathcal{O}_{\mathsf{y}}}$$

$\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}$ is a finite set of labeled directed edges. Each edge is either of the following:

$\dashrightarrow$ is an ontology edge corresponding to some subclass relationship specified in either ontology

$\longrightarrow$ is a mapping edge corresponding to some subclass or super class relationship specified in the mapping set

$\longleftrightarrow$ is a mapping edge corresponding to some equivalence relationship specified in the mapping set

Moreover,

$$\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} = \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \bigcup \mathbb{E}_{\mathcal{O}_{\mathsf{y}}} \bigcup \mathbb{E}$$

$\mathbb{E}$ is a finite set of *mapping edges* corresponding to the mapping relationships specified in the mapping set such that each edge contains a vertex from either ontology, that is, $\mathbb{E} = \{e_1^{\mathsf{x:y}}, e_2^{\mathsf{x:y}}, \dots\}$

$e_p^{\mathsf{x:y}}$ is a directed edge representing mapping relationship, such that,

$$\left(c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}\right) \Leftrightarrow \exists e_p^{\mathsf{x:y}} \in \mathbb{E}\colon \left(e_p^{\mathsf{x:y}} = v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}}\right)$$
$$\left(c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}\right) \Leftrightarrow \exists e_p^{\mathsf{x:y}} \in \mathbb{E}\colon \left(e_p^{\mathsf{x:y}} = v_j^{\mathsf{y}} \longrightarrow v_i^{\mathsf{x}}\right)$$
$$\left(c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}\right) \Leftrightarrow \exists e_p^{\mathsf{x:y}} \in \mathbb{E}\colon \left(e_p^{\mathsf{x:y}} = v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}}\right)$$

Property 3.1. *$\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ contains at most $\left(|\mathbb{C}_{\mathsf{x}}| + |\mathbb{C}_{\mathsf{y}}|\right)$ vertices and at most $\left(|\mathbb{R}_{\mathsf{x}}| + |\mathbb{R}_{\mathsf{y}}| + |\mathbb{M}_{\mathsf{x:y}}|\right)$ edges, that is,*

$$\left(|\mathbb{V}_{\mathcal{O}_{\mathsf{x}}}| + |\mathbb{V}_{\mathcal{O}_{\mathsf{y}}}|\right) = |\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}| \leq \left(|\mathbb{C}_{\mathsf{x}}| + |\mathbb{C}_{\mathsf{y}}|\right) = |\mathbb{C}|$$
$$\left(|\mathbb{E}_{\mathcal{O}_{\mathsf{x}}}| + |\mathbb{E}_{\mathcal{O}_{\mathsf{y}}}| + |\mathbb{E}|\right) = |\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}| \leq \left(|\mathbb{R}_{\mathsf{x}}| + |\mathbb{R}_{\mathsf{y}}| + |\mathbb{M}_{\mathsf{x:y}}|\right) = \left(|\mathbb{R}| + |\mathbb{M}|\right)$$

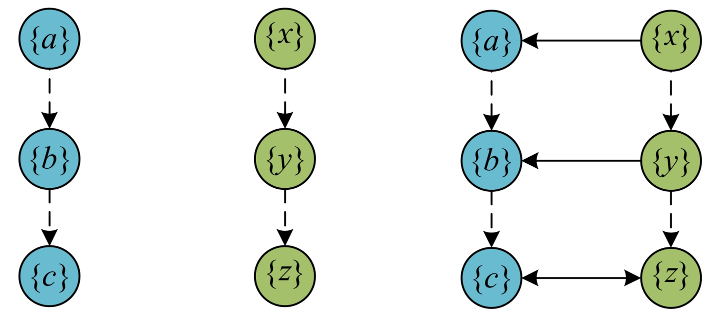
where $|\mathbb{C}| = |\mathbb{C}_{\mathsf{x}}| + |\mathbb{C}_{\mathsf{y}}|$, $|\mathbb{R}| = |\mathbb{R}_{\mathsf{x}}| + |\mathbb{R}_{\mathsf{y}}|$, and $|\mathbb{M}| = |\mathbb{M}_{\mathsf{x:y}}|$ as noted earlier in Section 2.5 and Property 2.1.

Example 3.1. For example, again consider the ontologies

$$\mathcal{O}_4\colon \left\langle \{a, b, c\}\,,\ \{c \prec b, b \prec a\} \right\rangle \text{ and } \mathcal{O}_5\colon \left\langle \{x, y, z\}\,,\ \{z \prec y, y \prec x\} \right\rangle$$

and a mapping set $\mathbb{M}_{4:5} = \{a \succ x, b \succ y, c \equiv z\}$. Its corresponding mapping graph $\mathcal{G}_{\mathbb{M}_{4:5}}$ is shown in Figure 3.1(c).

Note. We want to emphasize that we are using two different types of edges to distinguish the relationships that are specified in the ontologies and the relationships that are specified in the mapping set. Table 3.1 lists down how each relationship is being represented in the mapping graph.



(a) $\mathcal{G}_{\mathcal{O}_4}$ (b) $\mathcal{G}_{\mathcal{O}_5}$ (c) $\mathcal{G}_{\mathbb{M}_{4:5}}$

Figure 3.1: Example of Mapping Graph

Table 3.1: How are relationships represented in the mapping graph?

Relationship	In Mapping Graph	Edge Type
$c_i^{\mathsf{x}} \prec c_j^{\mathsf{x}}$	$v_i^{\mathsf{x}} \dashrightarrow v_j^{\mathsf{x}}$	Ontology edge
$c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}$	$v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}}$	Mapping edge
$c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}}$	$v_j^{\mathsf{y}} \longrightarrow v_i^{\mathsf{x}}$	Mapping edge
$c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}}$	$v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}}$	Mapping edge

3.3.1 Construction

Algorithm 3.2 shows an algorithm to construct mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ for given ontologies $\mathcal{O}_{\mathsf{x}} \colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}} \colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle$ and mapping set $\mathbb{M}_{\mathsf{x:y}}$. In the algorithm we first construct the ontology graphs for both the ontologies and then merge them by adding edges between the two graphs on the basis of the mapping relationships.

3.3.1.1 Complexity

In this construction, first we create two ontology graphs, for which the total running time is $O\left(|\mathbb{C}| + |\mathbb{R}|\right)$. Further, we iterate each mapping relation once. Hence, the total running time of the construction is $O\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)$.

3.3.1.2 Converting Mapping Graph to Ontology Graph

Each mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ for any given ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and mapping set $\mathbb{M}_{\mathsf{x:y}}$ can be converted to an ontology graph $\mathcal{G}_{\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}}$ for the combined ontology $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$. The conversion is

Algorithm 3.2 Constructing mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ for given ontologies $\mathcal{O}_{\mathsf{x}} \colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle$ and $\mathcal{O}_{\mathsf{y}} \colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle$ and mapping set $\mathbb{M}_{\mathsf{x:y}}$

```
Require: $\mathcal{O}_{\mathsf{x}} \colon \langle \mathbb{C}_{\mathsf{x}},\ \mathbb{R}_{\mathsf{x}} \rangle,\ \mathcal{O}_{\mathsf{y}} \colon \langle \mathbb{C}_{\mathsf{y}},\ \mathbb{R}_{\mathsf{y}} \rangle,\ \mathbb{M}_{\mathsf{x:y}}$
 1: Generate $\mathcal{G}_{\mathcal{O}_{\mathsf{x}}} \colon \langle \mathbb{V}_{\mathcal{O}_{\mathsf{x}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \rangle,\ \mathcal{G}_{\mathcal{O}_{\mathsf{y}}} \colon \left\langle \mathbb{V}_{\mathcal{O}_{\mathsf{y}}},\ \mathbb{E}_{\mathcal{O}_{\mathsf{y}}} \right\rangle$
 2: $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle,\ \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} := \mathbb{V}_{\mathcal{O}_{\mathsf{x}}} \bigcup \mathbb{V}_{\mathcal{O}_{\mathsf{y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} := \mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \bigcup \mathbb{E}_{\mathcal{O}_{\mathsf{y}}}$
 3: for all $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ do
 4:     $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} := \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \bigcup \left\{ v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}} \right\}$
 5: end for
 6: for all $c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ do
 7:     $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} := \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \bigcup \left\{ v_j^{\mathsf{y}} \longrightarrow v_i^{\mathsf{x}} \right\}$
 8: end for
 9: for all $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ do
10:     $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} := \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \bigcup \left\{ v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}} \right\}$
11: end for
```

quite simple. Firstly, we copy all the non-mapping edges and unidirectional mapping edges to the ontology graph. Then, we merge the vertices that are combined by the bidirectional edges. Algorithm 3.3 shows a simple algorithm to perform the same. The running time of the algorithm is $O\Big(|\mathbb{C}| + \big(|\mathbb{R}| + |\mathbb{M}|\big)^2\Big)$.

Theorem 3.1. *A merged ontology $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$ is consistent if its mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ is a DAG.*

Proof. This is obvious since the mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ can be converted to ontology graph $\mathcal{G}_{\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}}$ which in turn must be a DAG for the combined ontology $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$ to be consistent as per Theorem 2.1. □

3.3.2 Condition for Cycle

Representing the problem as a mapping graph, helps us understand a very important requirement for the inconsistency, described below.

Theorem 3.2. *Given any two consistent ontologies $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and a mapping set $\mathbb{M}_{\mathsf{x:y}}$, any cycle in the mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$ must contain at least two edges corresponding to the mapping relationships, that is, at least two edges in any cycle must belong to the set $\Big(\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \setminus \big(\mathbb{E}_{\mathcal{O}_{\mathsf{x}}} \bigcup \mathbb{E}_{\mathcal{O}_{\mathsf{y}}}\big)\Big)$. Moreover, those edges must be either of the following:*

Algorithm 3.3 Converting Mapping graph $\mathcal{G}_{\mathsf{M}_{\mathsf{x:y}}}$ to Ontology graph $\mathcal{G}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}}$

```
Require: $\mathcal{G}_{\mathsf{M}_{\mathsf{x:y}}} : \langle \mathbb{V}_{\mathsf{M}_{\mathsf{x:y}}}, \mathbb{E}_{\mathsf{M}_{\mathsf{x:y}}} \rangle$
1: $\mathcal{G}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} : \left\langle \mathbb{V}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}}, \mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} \right\rangle$, $\mathbb{V}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} := \mathbb{V}_{\mathsf{M}_{\mathsf{x:y}}}$, $\mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} := \emptyset$
2: for all $u \dashrightarrow v \in \mathbb{E}_{\mathsf{M}_{\mathsf{x:y}}}$ do
3:    $\mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} := \mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} \bigcup \{u \dashrightarrow v\}$
4: end for
5: for all $u \longrightarrow v \in \mathbb{E}_{\mathsf{M}_{\mathsf{x:y}}}$ do
6:    $\mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} := \mathbb{E}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} \bigcup \{u \longrightarrow v\}$
7: end for
8: for all $u \longleftrightarrow v \in \mathbb{E}_{\mathsf{M}_{\mathsf{x:y}}}$ do // Merge the vertices u and v into a single vertex //
9:    $w := u \bigcup v$
10:   for all $e \in \mathbb{E}_{\mathsf{M}_{\mathsf{x:y}}}$ do
11:      Replace $u$ or $v$ or both with $w$
12:   end for
13:   $\mathbb{V}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} := \left( \mathbb{V}_{\mathcal{O}_{\mathsf{M}_{\mathsf{x:y}}}} \bigcup \{w\} \right) \setminus \{u, v\}$
14: end for
```

- $v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}}$ *and* $v_l^{\mathsf{y}} \longrightarrow v_k^{\mathsf{x}}$

- $v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}}$ *and* $v_k^{\mathsf{x}} \longleftrightarrow v_l^{\mathsf{y}}$

- $v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}}$ *and* $v_l^{\mathsf{y}} \longrightarrow v_k^{\mathsf{x}}$

- $v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}}$ *and* $v_k^{\mathsf{x}} \longleftrightarrow v_l^{\mathsf{y}}$

Proof. The proof is simple. Since the given ontologies are consistent, they must be both DAGs by Theorem 2.1. Moreover, for the vertices in two DAGs to participate in a cycle, there must be a path that goes from each DAG to the other DAG, that is an edge from each DAG to the other DAG. The edges from one one DAG to other DAG are the ones coming due to the mapping set. Hence, at least two of the edges in any cycle must be the mapping edges as specified above. □

Remark. Now, we can observe that each cycle can in turn be represented as a chain of relationships which will lead to inconsistency in the ontology. Hence, any such chain would contain at least two mapping relationships which must be either of the following:

- $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}$ and $c_k^{\mathsf{x}} \succ c_l^{\mathsf{y}}$

- $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}$ and $c_k^{\mathsf{x}} \equiv c_l^{\mathsf{y}}$
- $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}}$ and $c_k^{\mathsf{x}} \succ c_l^{\mathsf{y}}$
- $c_i^{\mathsf{x}} \equiv c_j^{\mathsf{y}}$ and $c_k^{\mathsf{x}} \equiv c_l^{\mathsf{y}}$

3.4 Simple Naïve Approach

Now, we will discuss a very simple and straight-forward approach to compute a maximal consistent mapping subset. Recall that our problem is:
McM$_m$: Given consistent $\mathcal{O}_{\mathsf{x}}$ and $\mathcal{O}_{\mathsf{y}}$ and some $\mathbb{M}_{\mathsf{x:y}}$, find a subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \top$
2. $\forall r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}}: \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_{\mathsf{x:y}}}\right) = \bot\right)$ where $\mathbb{M}''_{\mathsf{x:y}} = \left(\mathbb{M}'_{\mathsf{x:y}} \bigcup \{r_p^{\mathsf{x:y}}\}\right)$

3.4.1 Approach

A very simple approach to identify a maximal subset would be to start with an empty subset $\mathbb{M}'_{\mathsf{x:y}}$. Now, we add one mapping at a time to this set, in decreasing order of their weights, as long as, the combined ontology $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ remains consistent. Once all mappings have been tried out, the subset at that time would be a maximal consistent mapping subset.

3.4.2 Algorithm

Algorithm 3.4 shows a simple implementation of the above approach.

3.4.3 Complexity

In this approach we add each mapping once and try to verify the consistency of the ontology. In order to check the consistency, we need to create the ontology graph for the combined ontology. We can optimize the process by creating an ontology graph for empty mapping set, and then forth, for each mapping, we just need to add one edge. Therefore, for each mapping, we need to check consistency. Hence, we have a running time of $O\big(|\mathbb{M}|\,(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)\big)$.

Algorithm 3.4 Simple Naïve Approach to solve McM$_m$

```
Require: $\mathcal{O}_{\mathsf{x}}$, $\mathcal{O}_{\mathsf{y}}$, $\mathbb{M}_{\mathsf{x:y}}$
1: $\mathbb{M}'_{\mathsf{x:y}} := \emptyset$
2: for all mapping $r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}}$ in decreasing order of weight where $p \in |\mathbb{M}_{\mathsf{x:y}}|$ do
3:     $\mathbb{M}'_{\mathsf{x:y}} = \mathbb{M}'_{\mathsf{x:y}} \bigcup \{r_p^{\mathsf{x:y}}\}$
4:     if consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \perp$ then
5:         $\mathbb{M}'_{\mathsf{x:y}} = \mathbb{M}'_{\mathsf{x:y}} \setminus \{r_p^{\mathsf{x:y}}\}$
6:     end if
7: end for
8: return $\mathbb{M}'_{\mathsf{x:y}}$
```

3.4.4 Correctness

Since we are looking at each mapping in the given order and a mapping is not added only when it causes inconsistency with the already added mappings. Hence, each mapping that was not added to the subset, is causing inconsistency with the other mappings already added in the subset, that is, no more mapping can be added to the subset. Therefore, the subset is maximal.

3.4.5 Approximation

This algorithm does not guarantee anything about the approximation it provides. The reason for this is simple. One mapping relationship with highest weight will be always added. Now, it is possible that this relationship is conflicting with all other relationships in the given mapping set. It is also possible that if this relationship were not added, then there was no inconsistency at all. Hence, the approximation simply depends on the order in which the mapping relationships are iterated.

3.5 Biased Approach

Our this approach builds up on the observation made in Section 3.3.2. We observed that there must be at least two mapping edges in each cycle and at least one mapping edge from each ontology graph to other ontology graph. That is, in terms of mapping relations, the two relations that may cause inconsistency must be $\prec$ and $\succ$ if none of the relation is $\equiv$. If all the relations are only $\prec$ or $\succ$ then there is no inconsistency. This observation gave us the idea for this approach.

3.5.1 Approach

The approach is simple. We first find out total weights of all the $\prec$ and $\succ$ mappings in the mapping set, say, $\omega^{\Sigma}(sub)$ and $\omega^{\Sigma}(sup)$ respectively. Now, whichever weight is higher, we add all those relationships to our mapping subset $\mathbb{M}'_{\mathsf{x:y}}$. At this state, the mapping set $\mathbb{M}'_{\mathsf{x:y}}$ does not cause any inconsistency in the combined ontology $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$, as discussed above. Now, we try to maximize $\mathbb{M}'_{\mathsf{x:y}}$, by adding one mapping out of all the remaining mappings, at a time to this $\mathbb{M}'_{\mathsf{x:y}}$ as long as, the combined ontology $\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}$ remains consistent. Once all the remaining mappings have been tried out, the subset $\mathbb{M}'_{\mathsf{x:y}}$ at that time would be a maximal set.

3.5.2 Algorithm

Algorithm 3.5 shows a simple implementation of the above approach.

Algorithm 3.5 Biased Approach to solve McM_m

```
Require: $\mathcal{O}_{\mathsf{x}}$, $\mathcal{O}_{\mathsf{y}}$, $\mathbb{M}_{\mathsf{x:y}}$
 1: Count $\omega^{\Sigma}(sub)$ := the sum of weights of all $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$
 2: Count $\omega^{\Sigma}(sup)$ := the sum of weights of all $c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$
 3: if $\omega^{\Sigma}(sub) > \omega^{\Sigma}(sup)$ then
 4:    $\mathbb{M}'_{\mathsf{x:y}}$ := subset of $\mathbb{M}_{\mathsf{x:y}}$ such that each mapping is of the form $c_i^{\mathsf{x}} \prec c_j^{\mathsf{y}}$
 5: else
 6:    $\mathbb{M}'_{\mathsf{x:y}}$ := subset of $\mathbb{M}_{\mathsf{x:y}}$ such that each mapping is of the form $c_i^{\mathsf{x}} \succ c_j^{\mathsf{y}}$
 7: end if
 8: $\mathbb{M}_{\mathsf{x:y}} := \mathbb{M}_{\mathsf{x:y}} \setminus \mathbb{M}'_{\mathsf{x:y}}$
 9: for all mapping $r_p^{\mathsf{x:y}} \in \mathbb{M}_{\mathsf{x:y}}$ in decreasing order of weight where $p \in |\mathbb{M}_{\mathsf{x:y}}|$ do
10:    $\mathbb{M}'_{\mathsf{x:y}} = \mathbb{M}'_{\mathsf{x:y}} \bigcup \{r_p^{\mathsf{x:y}}\}$
11:    if consistent $\left(\mathcal{O}_{\mathbb{M}'_{\mathsf{x:y}}}\right) = \perp$ then
12:       $\mathbb{M}'_{\mathsf{x:y}} = \mathbb{M}'_{\mathsf{x:y}} \setminus \{r_p^{\mathsf{x:y}}\}$
13:    end if
14: end for
```

3.5.3 Complexity

The only modification to this algorithm over Algorithm 3.4 is that before iterating over the mapping set, we compute the total weight of all subclass and super class relationships. This can be done with an additional running time of $O(|\mathbb{M}|)$ where $|\mathbb{M}|$ is the number of mappings. However, the overall complexity of the algorithm still remains same. Therefore, we have a running time of $O\big(|\mathbb{M}|\,(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)\big)$.

3.5.4 Correctness

At first we are adding only one kind of mappings, either $\prec$ or $\succ$. Now, these mapping alone do not cause any inconsistency. Then at each step when we try to add any of the remaining mapping, we add it only if it does not cause any inconsistency with the already added mappings. Hence, each mapping that was not added to the subset, is causing inconsistency with the other mappings already added in the subset, that is, no more mapping can be added to the subset. Therefore, the subset is maximal.

3.5.5 Approximation

The algorithm guarantees that the solution would contain at least half of the total weight of all non-equivalence mapping relationships. However, it does not guarantee anything about how many mappings out of the total mappings would be there in the solution. Further more, if the mapping set contains only the $\equiv$ relations, then there is again no guarantee of how many mappings would be present in the solution. Therefore, as discussed in Section 3.4.5, this algorithm also does not guarantee anything about the approximation achieved.

3.6 Graph-based Approach

As mentioned earlier in Section 3.3, Demetrescu and Finocchi identified a combinatorial algorithm for finding feedback arc set in a weighted directed graph [Demetrescu and Finocchi, 2003]. Given a weighted directed graph their algorithm can compute a minimal weight feedback arc set, such that the approximation ratio is bounded by the length of the longest simple cycle.

In order to use their algorithm we needed to identify a way to convert our problem into a feedback arc set in weighted directed graph problem. We can construct the mapping graph as per Section 3.3.1. Now, we need to convert the mapping graph to a standard weighted directed graph and in order to do that we must first identify a way to assign weights to the edges. We discuss various ways in which we can assign weights to edges in Section 3.6.4. Once we have computed the weights, we can replace different labeled edges with simple directed edges and use the the algorithm by Demetrescu and Finocchi to find out the minimal weight feedback arc set, as discussed in Section 3.6.5. Mapping relationships corresponding to all the mapping edges that are not in this feedback set form the maximal subset of relationships that can be added to combine the given ontologies without causing inconsistency, as discussed in Section 3.6.6.

3.6.1 Approach

Recall that the problem that we are trying to solve is: McM_m: Given consistent $\mathcal{O}_\mathsf{x}$ and $\mathcal{O}_\mathsf{y}$ and some $\mathbb{M}_\mathsf{x:y}$, find a subset $\mathbb{M}'_\mathsf{x:y} \subseteq \mathbb{M}_\mathsf{x:y}$ such that all the following are true:

1. consistent $\left(\mathcal{O}_{\mathbb{M}'_\mathsf{x:y}}\right) = \top$
2. $\forall r_p^\mathsf{x:y} \in \mathbb{M}_\mathsf{x:y} \setminus \mathbb{M}'_\mathsf{x:y} : \left(\text{consistent}\left(\mathcal{O}_{\mathbb{M}''_\mathsf{x:y}}\right) = \bot\right)$ where $\mathbb{M}''_\mathsf{x:y} = \left(\mathbb{M}'_\mathsf{x:y} \bigcup \{r_p^\mathsf{x:y}\}\right)$

We have multiple steps in this solution and in the sections below we will discuss each step individually. Here is the informal sketch of the solution:

1. **Input.** Input consistent ontologies $\mathcal{O}_\mathsf{x}: \langle \mathbb{C}_\mathsf{x}, \mathbb{R}_\mathsf{x}\rangle$ and $\mathcal{O}_\mathsf{y}: \langle \mathbb{C}_\mathsf{y}, \mathbb{R}_\mathsf{y}\rangle$ and weighted mapping set $\mathbb{M}_\mathsf{x:y}$.

2. **Mapping graph.** Compute mapping graph $\mathcal{G}_{\mathbb{M}_\mathsf{x:y}}: \left\langle \mathbb{V}_{\mathbb{M}_\mathsf{x:y}}, \mathbb{E}_{\mathbb{M}_\mathsf{x:y}}\right\rangle$ using $\mathcal{G}_{\mathcal{O}_\mathsf{x}}$ and $\mathcal{G}_{\mathcal{O}_\mathsf{y}}$ as per Algorithm 3.2 as discussed in Section 3.3.1.

3. **Mapping subgraph.** Compute a mapping subgraph $\mathcal{G}'_{\mathbb{M}_\mathsf{x:y}}: \left\langle \mathbb{V}'_{\mathbb{M}_\mathsf{x:y}}, \mathbb{E}'_{\mathbb{M}_\mathsf{x:y}}\right\rangle$ of $\mathcal{G}_{\mathbb{M}_\mathsf{x:y}}$ by removing all those vertices that can not appear in any cycles using Algorithm 3.6 as discussed in Section 3.6.3.

4. **Edge weighted directed graph.** Transform $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}}: \left\langle \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ into a weighted directed graph $G_{\mathcal{W}}: \langle V_{\mathcal{W}},\ E_{\mathcal{W}} \rangle$ by computing weight for each edge of $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}}$ using Algorithm 3.7 as discussed in Section 3.6.4.

5. **Minimal weight feedback arc set.** Compute the minimal weight feedback arc set $\mathbb{FAS} \subseteq E_{\mathcal{W}}$ for $G_{\mathcal{W}}$ using Algorithm 3.8 as discussed in Section 3.6.5. We will use the algorithm given by Demetrescu and Finocchi [2003].

6. **Maximal consistent mapping subset.** Compute the maximal consistent mapping subset $\mathbb{M}'_{\mathsf{x:y}} \subseteq \mathbb{M}_{\mathsf{x:y}}$ with the help of $\mathbb{FAS}$ using Algorithm 3.9 as discussed in Section 3.6.6.

7. **Output.** $\mathbb{M}'_{\mathsf{x:y}}$ is our solution

Example 3.2. Throughout this section we will follow a common example and perform each step as we proceed. Let us consider the following consistent ontologies

$$\mathcal{O}_6 : \left\langle \{a, b, c, d, e\},\ \{a \prec b, a \prec c, b \prec d, b \prec e, c \prec d\} \right\rangle \text{ and } \mathcal{O}_7 : \left\langle \{x, y, z\},\ \{x \prec y, x \prec z\} \right\rangle$$

and a corresponding weighted mapping set $\mathbb{M}_{4:5} = \{c \succ y, d \prec x, d \succ z\}$. For simplicity we assume that all the weights are unit weight.

Figure 3.2(a) and Figure 3.2(b) show the corresponding ontology graphs $\mathcal{G}_{\mathcal{O}_6}$ and $\mathcal{G}_{\mathcal{O}_7}$ respectively and Figure 3.2(c) shows the mapping graph $\mathcal{G}_{\mathbb{M}_{6:7}}$.

3.6.2 Useful functions

Before we start to discuss the remaining steps for the above approach, in the section, we will describe few functions that will help us to describe our approach more efficiently.

3.6.2.1 Function mapOut

At first, we define a function mapOut that, given a vertex, counts the total number of ways to reach the vertices in the other ontology, that is, count of the outgoing mapping edges. For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define function mapOut: $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \mathbb{N} \bigcup \{0\}$ as follows:

$$\text{mapOut}\left(v_i^{\mathsf{z}}\right) = \text{total number of all edges like } v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'} \text{ or } v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \text{ where } \mathsf{z} \neq \mathsf{z}' \in \{\mathsf{x}, \mathsf{y}\}$$

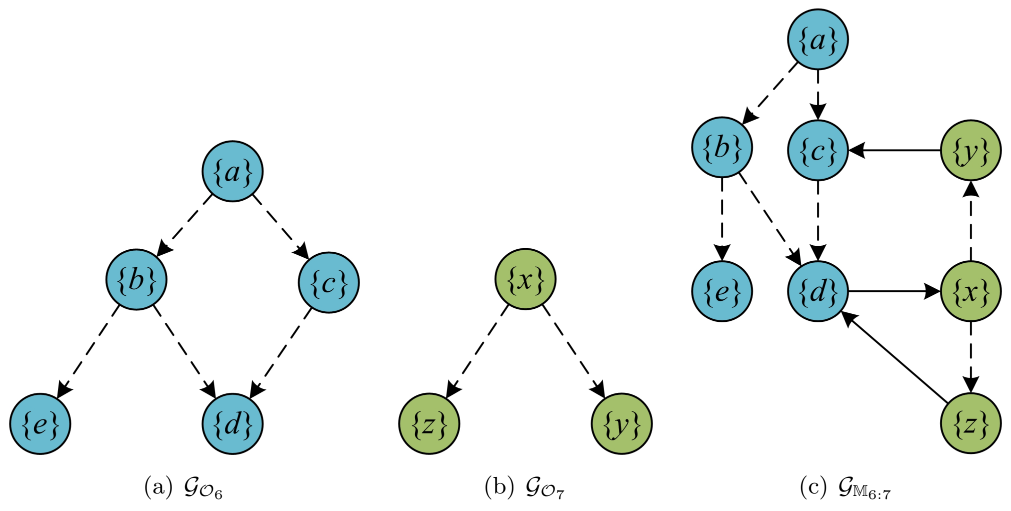

{a}
{b}
{c}
{e}
{d}
(a) $\mathcal{G}_{\mathcal{O}_6}$
{x}
{z}
{y}
(b) $\mathcal{G}_{\mathcal{O}_7}$
{a}
{b}
{c}
{y}
{e}
{d}
{x}
{z}
(c) $\mathcal{G}_{\mathbb{M}_{6:7}}$

Figure 3.2: Example ontology graphs and corresponding mapping graph for Graph-based Approach

It is easy to note that this function will have a runtime complexity of $O(|\mathbb{M}|)$ where $|\mathbb{M}|$ is the number of mapping relationships.

Example 3.3. For example, consider Figure 3.2(c). Here $\text{mapOut}(\{d\}) = 1$ since from $\{b\}$ there is only one outgoing mapping edge to $\{x\}$.

Moreover, $\text{mapOut}(\{x\}) = 0$ since from $\{x\}$ there is no outgoing mapping edge. Please note that, mapOut does not count the ontology edges outgoing to $\{y\}$ and $\{z\}$.

3.6.2.2 Function mapIn

Similar to the function mapOut, we define another function mapIn that, given a vertex, counts the number of all incoming mapping edges. For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define function $\text{mapIn}\colon \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \mathbb{N} \bigcup \{0\}$ as follows:

$$\text{mapIn}(v_i^{\mathsf{z}}) = \text{total number of all edges like } v_j^{\mathsf{z}'} \longrightarrow v_i^{\mathsf{z}} \text{ or } v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \text{ where } \mathsf{z} \neq \mathsf{z}' \in \{\mathsf{x}, \mathsf{y}\}$$

It is easy to note that this function will also have a runtime complexity of $O(|\mathbb{M}|)$ where $|\mathbb{M}|$ is the number of mapping relationships.

Example 3.4. For example, consider Figure 3.2(c). Here $\text{mapIn}(\{x\}) = 1$ since $\{x\}$ has only one incoming mapping edge from $\{d\}$.

Moreover, $\text{mapIn}(\{c\}) = 1$ too since $\{c\}$ has only one incoming mapping edge from $\{y\}$. Please note that, mapIn does not count the ontology edge incoming from $\{a\}$.

3.6.2.3 Function ontReach

Now, we define a reachability function. Given two vertices, belonging to the same ontology graph, this function computes if it is possible to reach one vertex from the other or not? If the two vertices are same or both are from different ontology graphs, then this function returns *false*. This function will be useful in defining the next two useful functions. For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define function $\text{ontReach}\colon \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \times \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \{\top, \bot\}$ as follows:

$$\text{ontReach}\left(v_i^{\mathsf{z}}, v_j^{\mathsf{z}}\right) = \begin{cases} \top & \text{if } v_i^{\mathsf{z}} \neq v_j^{\mathsf{z}} \text{ and } \exists v_1^{\mathsf{z}}, v_2^{\mathsf{z}}, \ldots, v_n^{\mathsf{z}} \in \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}: \\ & \left(v_i^{\mathsf{z}} \dashrightarrow v_1^{\mathsf{z}} \in \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}\right) \wedge \left(v_1^{\mathsf{z}} \dashrightarrow v_2^{\mathsf{z}} \in \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}\right) \wedge \cdots \wedge \left(v_n^{\mathsf{z}} \dashrightarrow v_j^{\mathsf{z}} \in \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}\right) \\ \bot & \text{otherwise} \end{cases}$$

where $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$.

Informally, $\text{ontReach}\left(v_i^{\mathsf{z}}, v_j^{\mathsf{z}}\right) = \top$, if it is possible to reach to v_j^{z} starting from v_i^{z} following only the non-mapping edges. We would also like to note that this function can be computed easily using a modified version of standard depth-first search where we ignore all the non-mapping edges. It turns out that we can compute it with a runtime complexity of $O\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)$.

Example 3.5. For example, consider Figure 3.2(c). Here $\text{ontReach}(\{a\}, \{e\}) = \top$ since it is possible to reach $\{e\}$ from $\{a\}$ following only the non-mapping edges.

Moreover, $\text{ontReach}(\{b\}, \{c\}) = \bot$ since it is not possible to reach $\{c\}$ from $\{b\}$ following only the non-mapping edges. Please note that, ontReach ignores the path through the mapping edge $d \longrightarrow x$ and $y \longrightarrow c$.

Note. We will like to note that this function results into *false* if both the input vertices are same or both of them actually belonged to different ontology graphs, that is, $\text{ontReach}(v_i^{\mathsf{x}}, v_i^{\mathsf{x}})$, $\text{ontReach}(v_i^{\mathsf{y}}, v_i^{\mathsf{y}})$, $\text{ontReach}\left(v_j^{\mathsf{y}}, v_i^{\mathsf{x}}\right)$, and $\text{ontReach}\left(v_j^{\mathsf{x}}, v_i^{\mathsf{y}}\right)$ all result into $\bot$.

3.6.2.4 Function ontAnc

Now, we will define a function to find out all the ancestors of a given vertex. This function is different from the standard functions that detect ancestors in a way like ontReach, this function also ignores paths through mapping edges. Given a vertex, this function computes a set of all those vertices belonging to the same ontology graph, that are reachable from this vertex through non-mapping edges (*ancestors* within same ontology). For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define function ontAnc: $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \wp\left(\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}\right)$ where $\wp\left(\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}\right)$ is the power set of $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$, as follows:

$$\text{ontAnc}\left(v_i^{\mathsf{z}}\right) = V \text{ such that } \forall v_j^{\mathsf{z}} \in V\colon \text{ ontReach}\left(v_i^{\mathsf{z}}, v_j^{\mathsf{z}}\right) = \top \text{ and } \forall v_k^{\mathsf{z}} \in \overline{V}\colon \text{ ontReach}\left(v_i^{\mathsf{z}}, v_k^{\mathsf{z}}\right) = \bot$$

where $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$ and $\overline{V} = \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \setminus V$

We note that ontAnc$\left(v_i^{\mathsf{z}}\right)$ gives the set of all those vertices that are reachable from v_i^{z} through non-mapping edges. We refer these vertices as *ancestors* of v_i^{z}. We would also like to note that if we compute this function the way we have defined it, then we can do this with a runtime complexity of $O\Big(|\mathbb{C}|\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)\Big)$. However, rather than computing it like this, we can do this simply by modifying the standard depth-first search algorithm and thereby attain a runtime complexity of $O\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)$.

Example 3.6. Consider Figure 3.2(c). Here ontAnc$\left(\{a\}\right) = \{\{b\}, \{c\}, \{d\}, \{e\}\}$ since it is possible to reach all $\{b\}$, $\{c\}$, $\{d\}$ and $\{e\}$ from $\{a\}$ following only the non-mapping edges. We note ontAnc does not include $\{x\}$ in this set since it is not possible to reach $\{x\}$ from $\{a\}$ following only the non-mapping edges.

3.6.2.5 Function ontDesc

Similar to ontAnc, we define another function ontDesc that, given a vertex, computes a set of all those vertices belonging to the same ontology graph, from where this vertex is reachable through non-mapping edges (*descendants* within same ontology). For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define function ontDesc: $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \wp\left(\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}\right)$ where $\wp\left(\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}\right)$ is the power set of $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$, as follows:

$$\text{ontDesc}\left(v_i^{\mathsf{z}}\right) = \mathbb{V} \text{ such that } \forall v_j^{\mathsf{z}} \in V\colon\ \text{ontReach}\left(v_j^{\mathsf{z}}, v_i^{\mathsf{z}}\right) = \top$$
$$\text{and } \forall v_k^{\mathsf{z}} \in \overline{V}\colon\ \text{ontReach}\left(v_k^{\mathsf{z}}, v_i^{\mathsf{z}}\right) = \bot \text{ where } \mathsf{z} \in \{\mathsf{x}, \mathsf{y}\} \text{ and } \overline{V} = \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \setminus V$$

Moreover, like ontAnc, ontAnc will also have a runtime complexity of $O\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)$.

Example 3.7. For example, consider Figure 3.2(c). Here $\text{ontDesc}\left(\{d\}\right) = \{\{a\}, \{b\}, \{c\}\}$ since it is possible to reach both $\{d\}$ from all $\{a\}$, $\{b\}$ and $\{c\}$ following only the non-mapping edges. We note ontAnc does not include $\{z\}$ in this set since it is not possible to reach $\{d\}$ from $\{z\}$ following only the non-mapping edges.

3.6.3 How to compute Subgraph?

Now, we can describe our approach further. The next step after computing the mapping graph, is to compute the subgraph. Since the performance of graph problems is usually dependent on the vertices and edges in the graph, we think that if we can efficiently remove the vertices that are guaranteed not to participate in any cycle, then the runtime performance of our algorithm will be much better (even though it may not improve the worst case runtime complexity). We want to note here that this step is optional, and our solution will work even if we skip this step.

In this step, given a mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}\colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}, \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$, we try to find out all those vertices $v_i^{\mathsf{z}} \in \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$ where $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$ that cannot participate in any cycle at all and remove them from the graph. It is obvious that if a vertex has no incoming or no outgoing edge, then it cannot participate in a cycle. For a vertex to participate in any cycle, it must have both incoming and outgoing edges. However, in our case, we can prove a stronger condition. We do that in the following theorem.

Theorem 3.3 (Condition for a vertex to *possibly* participate in some cycle)**.** *Given a mapping graph* $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}\colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}, \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$*, a vertex* $v_i^{\mathsf{z}} \in \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$ *where* $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$ *may participate in a cycle only if at least one of the following conditions is true:*

1. $\left(\text{mapOut}\left(v_i^{\mathsf{z}}\right) > 0\right) \wedge \left(\text{mapIn}\left(v_i^{\mathsf{z}}\right) > 0\right)$

2. $(\text{mapIn}(v_i^z) > 0) \wedge \left(\exists v_j^z \in \text{ontAnc}(v_i^z) :\ \text{mapOut}\left(v_j^z\right) > 0\right)$

3. $(\text{mapOut}(v_i^z) > 0) \wedge (\exists v_k^z \in \text{ontDesc}(v_i^z) :\ \text{mapIn}(v_k^z) > 0)$

4. $\left(\exists v_j^z \in \text{ontAnc}(v_i^z) :\ \text{mapOut}\left(v_j^z\right) > 0\right) \wedge (\exists v_k^z \in \text{ontDesc}(v_i^z) :\ \text{mapIn}(v_k^z) > 0)$

Alternatively, a vertex $v_i^z \in \mathbb{V}_{\mathbb{M}_{x:y}}$ *may participate in a cycle only if*

$$\left(\text{mapIn}(v_i^z) > 0 \vee (\exists v_k^z \in \text{ontDesc}(v_i^z) :\ \text{mapIn}(v_k^z) > 0)\right)$$
$$\wedge \left(\text{mapOut}(v_i^z) > 0 \vee (\exists v_j^z \in \text{ontAnc}(v_i^z) :\ \text{mapOut}\left(v_j^z\right) > 0)\right) \quad (3.1)$$

In simple words, a vertex in a mapping graph can participate in a cycle only if either it or one of its descendant has an incoming mapping edge and either it or one of its ancestors has an outgoing mapping edge.

Proof. Without loss of generality, let us assume by contradiction that there exists some vertex v_i^x that participates in a cycle even though Equation (3.1) is false, that is,

$$\neg\Big(\left(\text{mapIn}(v_i^z) > 0 \vee (\exists v_k^z \in \text{ontDesc}(v_i^z) :\ \text{mapIn}(v_k^z) > 0)\right)$$
$$\wedge \left(\text{mapOut}(v_i^z) > 0 \vee (\exists v_j^z \in \text{ontAnc}(v_i^z) :\ \text{mapOut}\left(v_j^z\right) > 0)\right)\Big)$$

$$= \neg\left(\text{mapIn}(v_i^x) > 0 \vee (\exists v_k^x \in \text{ontDesc}(v_i^x) :\ \text{mapIn}(v_k^x) > 0)\right)$$
$$\vee \neg\left(\text{mapOut}(v_i^x) > 0 \vee (\exists v_j^x \in \text{ontAnc}(v_i^x) :\ \text{mapOut}\left(v_j^x\right) > 0)\right)$$

$$= \left(\text{mapIn}(v_i^x) \not> 0 \wedge (\nexists v_k^x \in \text{ontDesc}(v_i^x) :\ \text{mapIn}(v_k^x) > 0)\right)$$
$$\vee \left(\text{mapOut}(v_i^x) \not> 0 \wedge (\nexists v_j^x \in \text{ontAnc}(v_i^x) :\ \text{mapOut}\left(v_j^x\right) > 0)\right)$$

Hence, we have following two conditions:

$$\text{mapIn}(v_i^x) \not> 0 \wedge (\nexists v_k^x \in \text{ontDesc}(v_i^x) :\ \text{mapIn}(v_k^x) > 0) \quad (3.2)$$
$$\text{mapOut}(v_i^x) \not> 0 \wedge (\nexists v_j^x \in \text{ontAnc}(v_i^x) :\ \text{mapOut}\left(v_j^x\right) > 0) \quad (3.3)$$

Now, at least one out of Equation (3.2) or Equation (3.3) must be *true* for v_i^x to participate in a cycle. However, if Equation (3.2) is true, then it means that neither v_i^x nor any of its descendants has an incoming mapping edge. This means that v_i^{x} is not reachable via the mapping edges. As per Theorem 3.2 each cycle must contain mapping edges. Therefore, this is contradiction and hence Equation (3.2) must be *false*. Alternatively, if Equation (3.3) is *true*, then it means that neither v_i^{x} nor any of its ancestors have outgoing mapping edge. This means that none of the vertices reachable from v_i^{x} can be reached via the mapping edges. Again, we have a contradiction and hence Equation (3.3) must be *false* too. Since, none of the conditions are *true*, our assumption is not valid and therefore, Equation (3.1) must hold for v_i^{x} to participate in a cycle. □

Remark. We would like to note here that Theorem 3.3 does not guarantee that if any of the conditions is *true* for some vertex then that vertex will surely participate in some cycle. It only guarantees that a vertex will not participate in a cycle if none of the conditions are *true*. We can use this theorem to define the next function.

3.6.3.1 Function mayInCycle

For any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$, we define a function mayInCycle: $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \{\top, \bot\}$ as follows:

$$\mathrm{mayInCycle}\left(v_i^{\mathsf{z}}\right) = \begin{cases} \top & \text{if } \Big(\mathrm{mapIn}\left(v_i^{\mathsf{z}}\right) > 0 \vee \left(\exists v_k^{\mathsf{z}} \in \mathrm{ontDesc}\left(v_i^{\mathsf{z}}\right) :\ \mathrm{mapIn}\left(v_k^{\mathsf{z}}\right) > 0\right)\Big) \\ & \wedge\Big(\mathrm{mapOut}\left(v_i^{\mathsf{z}}\right) > 0 \vee \left(\exists v_j^{\mathsf{z}} \in \mathrm{ontAnc}\left(v_i^{\mathsf{z}}\right) :\ \mathrm{mapOut}\left(v_j^{\mathsf{z}}\right) > 0\right)\Big) \\ \bot & \text{otherwise} \end{cases}$$

The number of ancestors or descendants for a vertex can be at most $|\mathbb{C}|$. Hence, this function can be computed with a runtime complexity of $O\Big(|\mathbb{C}|\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)\Big)$.

3.6.3.2 Algorithm

Now, we can use the function mayInCycle to write an algorithm for this step. Algorithm 3.6 shows how to compute a subgraph $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}} : \left\langle \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ of $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} : \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$.

Since at most we have $|\mathbb{C}|$ number of vertices, this algorithm will therefore have a runtime complexity of $O\Big(|\mathbb{C}|^2\left(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|\right)\Big)$.

Example 3.8. For example, consider the mapping graph $\mathcal{G}_{\mathbb{M}_{5:6}}$ shown in Figure 3.2(c). Its corresponding subgraph $\mathcal{G}'_{\mathbb{M}_{5:6}}$ is shown in Figure 3.3.

Algorithm 3.6 Computing subgraph $\mathcal{G}'_{\mathbb{M}_{x:y}}\colon \left\langle \mathbb{V}'_{\mathbb{M}_{x:y}},\ \mathbb{E}'_{\mathbb{M}_{x:y}} \right\rangle$ of $\mathcal{G}_{\mathbb{M}_{x:y}}\colon \langle \mathbb{V}_{\mathbb{M}_{x:y}},\ \mathbb{E}_{\mathbb{M}_{x:y}} \rangle$

```
Require: $\mathcal{G}_{\mathbb{M}_{x:y}}\colon \langle \mathbb{V}_{\mathbb{M}_{x:y}},\ \mathbb{E}_{\mathbb{M}_{x:y}} \rangle$
1: $\mathcal{G}'_{\mathbb{M}_{x:y}}\colon \left\langle \mathbb{V}'_{\mathbb{M}_{x:y}},\ \mathbb{E}'_{\mathbb{M}_{x:y}} \right\rangle$, $\mathbb{V}'_{\mathbb{M}_{x:y}} := \mathbb{V}_{\mathbb{M}_{x:y}}$, $\mathbb{E}'_{\mathbb{M}_{x:y}} := \mathbb{E}_{\mathbb{M}_{x:y}}$
2: for all $v_i^{x} \in \mathbb{V}_{\mathbb{M}_{x:y}}$ do
3:    if mayInCycle$(v_i^{x}) = \perp$ then
4:       $\mathbb{V}'_{\mathbb{M}_{x:y}} := \mathbb{V}'_{\mathbb{M}_{x:y}} \setminus \{v_i^{x}\}$ // also remove corresponding edges //
5:    end if
6: end for
7: return $\mathcal{G}'_{\mathbb{M}_{x:y}}$
```

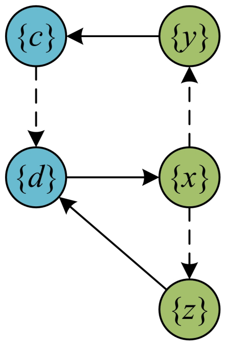


Figure 3.3: Example for mapping subgraph

3.6.4 How to compute Weights for Edges?

As discussed earlier in Section 3.6, in order to use the algorithm by Demetrescu and Finocchi, we must first compute weights for the edges in our mapping graph and then convert it to a standard edge weighted directed graph. The latter step is simple as we only need to remove the labels from the edges and we can perform this at the same time while computing the weights as shown later in Algorithm 3.7. Hence, here we will first discuss the various ways in which we can assign weights to the edges. Let us say that, for some edge weighted directed graph $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}} \rangle$, we are interested in initializing the function

$$\text{weight}\colon E_{\mathcal{W}} \longrightarrow \mathbb{N} \bigcup \{\infty\}$$

.

First, we note that this Demetrescu and Finocchi's algorithm works by removing edges with the minimum weight. We also know that, under no circumstance, shall we be removing the non-mapping relationships, that is, we do not want to remove the non-mapping edges. Hence, it is clear that the non-mapping edges must be all assigned *infinite* weight. Now, below we will discuss only about how we can assign weights to the mapping edges.

One very simple and straight forward approach is that we can assign *constant* weight to all those edges. This way all the edges that are causing cycles will get removed with equal preference.

Another simple approach is to use the user specified mapping weights. If we have ω for all the mappings then we may also choose the assign the same weights to each mapping edge. This way the edges causing cycles will be removed as per the user preference.

However, we know that some edges may be causing more cycles than some other edges and if we first remove the edges that are participating in more cycles then we may get a larger solution set. That is, we try to assign weights to the mapping edges such that an edge that appears in more number of cycles, gets removed before an edge that appears in less number of cycles. To achieve this, we first identify the number of cycles that each mapping edge can participate in, that is, we are interested in implementing some function numCycle: $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \setminus \left(\mathbb{E}_{\mathcal{O}_\mathsf{x}} \bigcup \mathbb{E}_{\mathcal{O}_\mathsf{y}}\right) \longrightarrow \mathbb{N}$ such that,

$$\text{numCycle}\left(e_p^{\mathsf{x:y}}\right) = \text{ Number of cycles in which edge } e_p^{\mathsf{x:y}} \text{ participates}$$

Once, we have this information, we can assign the lowest weight to the edges which participate in most number of cycles and maximum weight to the edges which participate in least number of cycles.

One approach to implement this is by identifying all the elementary cycles in the graph and based on that compute the number of cycles in which each mapping edge participates. However, it is not possible to compute it efficiently. One of the efficient algorithms that can be used is time bounded by $O\Big(\big(|\mathbb{V}| + |\mathbb{E}|\big)\big(c+1\big)\Big)$ and space bounded by $(|\mathbb{V}| + |\mathbb{E}|)$ where c is the number of elementary cycles in the graph [Johnson, 1975]. Since the number of elementary cycles can be really huge, we cannot compute this in polynomial time. Therefore, rather than calculating the exact number of cycles, we try to estimate it as shown below.

3.6.4.1 Heuristically Estimating the Number of Cycles for an Edge

Earlier in Theorem 3.2 we proved that each cycle in a mapping graph must contain at least two mapping edges so that there is both a way to enter and exit each ontology graph once. Therefore, we can estimate the number of cycles a mapping edge can participate in by finding out the number of ways it is possible to enter the edge from the one ontology graph and the number of ways it is possible to leave into the other ontology graph. That is,

$$\begin{aligned}\text{numCycle}\left(u \longrightarrow v\right) &= \Big(\text{Number of ways to enter } u \text{ from other ontology}\Big)\\ &\times \Big(\text{Number of ways to leave } v \text{ into other ontology}\Big)\\ \text{numCycle}\left(u \longleftrightarrow v\right) &= \text{numCycle}\left(u \longrightarrow v\right) + \text{numCycle}\left(v \longrightarrow u\right)\end{aligned}$$

Therefore, for each mapping edge we can estimate the maximum number of possible cycles that it can participate in using the following calculations:

$$\begin{aligned}\text{numCycle}\left(v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'}\right) &= \left(\text{mapIn}\left(v_i^{\mathsf{z}}\right) + \sum_{v_k^{\mathsf{z}} \in \text{ontDesc}\left(v_i^{\mathsf{z}}\right)} \text{ontDescPath}\left(v_k^{\mathsf{z}}, v_i^{\mathsf{z}}\right) \text{mapIn}\left(v_k^{\mathsf{z}}\right)\right)\\ &\times \left(\text{mapOut}\left(v_j^{\mathsf{z}'}\right) + \sum_{v_k^{\mathsf{z}'} \in \text{ontAnc}\left(v_j^{\mathsf{z}'}\right)} \text{ontAncPath}\left(v_j^{\mathsf{z}'}, v_k^{\mathsf{z}'}\right) \text{mapOut}\left(v_k^{\mathsf{z}'}\right)\right)\\ \text{numCycle}\left(v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'}\right) &= \text{numCycle}\left(v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'}\right) + \text{numCycle}\left(v_j^{\mathsf{z}'} \longrightarrow v_i^{\mathsf{z}}\right)\end{aligned}$$

where $\mathsf{z} \neq \mathsf{z}' \in \{\mathsf{x}, \mathsf{y}\}$ and for any mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} : \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}, \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ we define the functions $\text{ontAncPath} \colon \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \times \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \mathbb{N} \bigcup \{0\}$ and $\text{ontDescPath} \colon \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \times \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} \longrightarrow \mathbb{N} \bigcup \{0\}$ as follows:

$$\text{ontAncPath}\left(v_d^{\mathsf{z}}, v_a^{\mathsf{z}}\right) = \begin{cases} 0 & \text{if } v_a^{\mathsf{z}} \notin \text{ontAnc}\left(v_d^{\mathsf{z}}\right) \\ \left|\text{ontAnc}\left(v_d^{\mathsf{z}}\right)\right| & \text{otherwise — overestimation} \end{cases}$$

$$\text{ontDescPath}\left(v_d^{\mathsf{z}}, v_a^{\mathsf{z}}\right) = \begin{cases} 0 & \text{if } v_d^{\mathsf{z}} \notin \text{ontDesc}\left(v_a^{\mathsf{z}}\right) \\ \left|\text{ontDesc}\left(v_a^{\mathsf{z}}\right)\right| & \text{otherwise — overestimation} \end{cases}$$

In all other cases numCycle results into 0.

Ideally, with $\text{ontAncPath}\left(v_d^{\mathsf{z}}, v_a^{\mathsf{z}}\right)$ we are interested in computing all the paths from vertex v_d^{z} to its ancestor v_a^{z} and with $\text{ontDescPath}\left(v_d^{\mathsf{z}}, v_a^{\mathsf{z}}\right)$ we are interested in computing all the paths to vertex v_a^{z} from its descendant v_d^{z}. However, since we cannot compute that efficiently, we simply overestimate it to the actual number of ancestors and descendants respectively since the number of path cannot be greater than that number.

Moreover, we can observe that since maximum value for ancestors and descendants is less than $|\mathbb{C}|$, this function can be computed with a runtime complexity of $O\Big(|\mathbb{C}|\left(|\mathbb{C}|+|\mathbb{R}|+|\mathbb{M}|\right)\Big)$.

3.6.4.2 Algorithm

We can therefore assign the weight to each edge define the weights for each edge as a function of both the user specified weight and the number of cycles in which that edge may participate such that the edge with higher weight and that may participate in fewer cycles gets precedence over an edge with less weight and that may participate in more cycles. We use a possible approach in the next algorithm. Algorithm 3.7 shows an algorithm to convert the mapping subgraph $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}}\colon \left\langle \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ into normal weighted directed graph $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}}\rangle$. In the algorithm we assign ∞ weight to each non-mapping edge. Then for each mapping edge $e_p^{\mathsf{x:y}}$, we compute the value of numCycle $\left(e_p^{\mathsf{x:y}}\right)$ and then assume *max* to be some number greater than the maximum value of all numCycle $\left(e_p^{\mathsf{x:y}}\right)$. Then each mapping edge $e_p^{\mathsf{x:y}}$ is assigned a weight of $max - \text{numCycle}\left(e_p^{\mathsf{x:y}}\right) + \omega\left(e_p^{\mathsf{x:y}}\right)$.

Since we are computing numCycle for each mapping edge, therefore, this algorithm has a runtime complexity of $O\Big(|\mathbb{C}|\,|\mathbb{M}|\left(|\mathbb{C}|+|\mathbb{R}|+|\mathbb{M}|\right)\Big)$.

Example 3.9. For example, consider the mapping subgraph $\mathcal{G}'_{\mathbb{M}_{5:6}}$ shown in Figure 3.3. We convert it to an edge weighted directed graph $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}}\rangle$ as shown in Figure 3.4. Here to compute the weights we assumed $max = 3$.

3.6.5 How to compute Minimum Weight Feedback Arc Set?

As discussed earlier, Demetrescu and Finocchi gave an approximation algorithm for finding feedback arc set in a weighted directed graph [Demetrescu and Finocchi, 2003]. Given a weighted directed graph their algorithm computes a minimal weight feedback arc set. This is a set of edges with minimal weight and when these edges are removed from the graph, the

Algorithm 3.7 Creating $G_{\mathcal{W}} \colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}} \rangle$ from $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$

```
Require: $\mathcal{G}'_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$
 1: $G_{\mathcal{W}} \colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}} \rangle$, $V_{\mathcal{W}} := \mathbb{V}'_{\mathbb{M}_{\mathsf{x:y}}}$, $E_{\mathcal{W}} := \emptyset$
 2: for all $v_i^{\mathsf{z}} \dashrightarrow v_j^{\mathsf{z}} \in \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}}$ where $\mathsf{z} \in \{\mathsf{x}, \mathsf{y}\}$ do
 3:     $E_{\mathcal{W}} := E_{\mathcal{W}} \bigcup \left\{ v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}} \right\}$
 4:     weight $\left( v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}} \right) := \infty$
 5: end for
 6: Let $max$ be some number greater than maximum value of numCycle
 7: for all $v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'} \in \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}}$ where $\mathsf{z} \neq \mathsf{z}' \in \{\mathsf{x}, \mathsf{y}\}$ do
 8:     $E_{\mathcal{W}} := E_{\mathcal{W}} \bigcup \left\{ v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'} \right\}$
 9:     weight $\left( v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'} \right) := max - $ numCycle $\left( v_i^{\mathsf{z}} \longrightarrow v_j^{\mathsf{z}'} \right) + \omega \left( c_i^{\mathsf{z}} \prec c_j^{\mathsf{z}'} \right)$
10: end for
11: for all $v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \in \mathbb{E}'_{\mathbb{M}_{\mathsf{x:y}}}$ where $\mathsf{z} \neq \mathsf{z}' \in \{\mathsf{x}, \mathsf{y}\}$ do
12:     $E_{\mathcal{W}} := E_{\mathcal{W}} \bigcup \left\{ v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \right\}$
13:     weight $\left( v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \right) := max - $ numCycle $\left( v_i^{\mathsf{z}} \longleftrightarrow v_j^{\mathsf{z}'} \right) + \omega \left( c_i^{\mathsf{z}} \equiv c_j^{\mathsf{z}'} \right)$
14: end for
```

graph is cycle free. We have now converted our mapping graph $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}} \colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$ to a edge weighted directed graph $G_{\mathcal{W}} \colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}} \rangle$. Therefore, we can directly use their algorithm. We present their algorithm in our context in Algorithm 3.8.

The algorithm works in two phases. At first, it tries to identify a simple cycle C in the graph and remove all the edges with minimum weight ϵ in that cycle. The weight of all the other edges in that cycle C is reduced by ϵ. This step is repeated until their are no more cycles in the graph. Now, the set of all the removed edges is a feedback arc set, though not necessarily minimal since some of the remove edges may not be participating in the same cycle and hence it may be possible to add some of them back to the graph. Hence, in the next phase, it tries to add each edge in the feedback arc set to the graph, in an arbitrary order, as long as that edge does not lead to any cycle in the graph. The feedback arc set at the end of this phase is a minimal.

As computed by Demetrescu and Finocchi, the worst runtime complexity of this algorithm is $O\left(|\mathbb{V}|\,|\mathbb{E}|\right)$ which in our case turns out to be $O\Big(|\mathbb{C}|\,\left(|\mathbb{R}| + |\mathbb{M}|\right)\Big)$. Moreover this algorithm

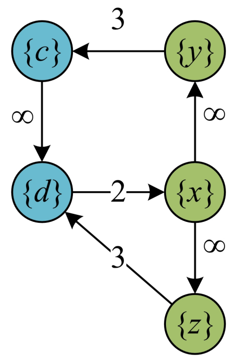


Figure 3.4: Example for edge weighted directed graph

guarantees an approximation ratio bounded by the length of the longest simple cycle in the graph given in terms of the number of edges independent of their weight.

Example 3.10. For example, consider the edge weighted directed graph $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}}\rangle$ shown in Figure 3.4. The minimal feedback arc set $\mathbb{FAS}$ for this is $\Big\{\{d\} \longrightarrow \{x\}\Big\}$.

3.6.6 How to get Mapping Set from Feedback Arc Set?

A successful execution of Algorithm 3.8 will give us a minimal set of edges that participated in some cycle in the graph. We can easily observe that these edges are all going to be the mapping edges since the non-mapping edges had a weight of ∞ and hence, they will not get removed at all. Further more, all mapping edges correspond to mapping relationships. So, of we remove all these mapping relationships from the original mapping set then the remaining subset of mapping will contain all those mappings that are not corresponding to any cycle in the mapping graph, that is, they are not causing any inconsistency in the combined ontology. Algorithm 3.9 shows a sample implementation of this process.

First we are converting the edges in feedback arc set to mapping relationships, which can be done in $O\left(|\mathbb{C}|^2\,|\mathbb{M}|\right)$. And then we remove all these mapping relationships from the original mapping set, which can be done in $O\left(|\mathbb{M}|^2\right)$. Hence, for this algorithm, we get a runtime complexity of $O\Big(|\mathbb{M}|\,\big(|\mathbb{M}| + |\mathbb{C}|^2\big)\Big)$.

Example 3.11. For example, consider the $\mathbb{FAS} = \Big\{\{d\} \longrightarrow \{x\}\Big\}$ computed earlier and

Algorithm 3.8 Computing Minimal Feedback Arc Set of $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}}\rangle$ using Demetrescu and Finocchi's algorithm

```
Require: $G_{\mathcal{W}}\colon \langle V_{\mathcal{W}},\ E_{\mathcal{W}}\rangle$
 1: $\mathbb{FAS} := \emptyset$
 2: while $\langle V_{\mathcal{W}}, E_{\mathcal{W}} \setminus \mathbb{FAS}\rangle$ is not acyclic do
 3:   Let $C$ be a simple cycle in $E_{\mathcal{W}} \setminus \mathbb{FAS}$
 4:   Let $\epsilon$ be the minimum weight of all the edges in $C$
 5:   for all edge $e \in C$ do
 6:     $\text{weight}(e) := \text{weight}(e) - \epsilon$
 7:     if $\text{weight}(e) = 0$ then
 8:       $\mathbb{FAS} := \mathbb{FAS} \bigcup \{e\}$
 9:     end if
10:   end for
11: end while
12: for all $e \in \mathbb{FAS}$ do
13:   if $\langle V_{\mathcal{W}}, (E_{\mathcal{W}} \setminus \mathbb{FAS}) \bigcup \{e\}\rangle$ is acyclic then
14:     $\mathbb{FAS} := \mathbb{FAS} \setminus \{e\}$
15:   end if
16: end for
17: return $\mathbb{FAS}$
```

original mapping set $\mathbb{M}_{5:6} = \{c \succ y, d \prec x, d \succ z\}$. Algorithm 3.9 will thus compute the maximal mapping subset $\mathbb{M}'_{5:6} = \{c \succ y, d \succ z\}$.

3.6.7 Complexity

The complexity of various steps is as follows:

- **Generating Mapping graph.** $O(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)$
- **Generating Mapping subgraph.** $O\Big(|\mathbb{C}|^2 (|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)\Big)$
- **Generating Edge weighted directed graph.** $O\Big(|\mathbb{C}|\,|\mathbb{M}| (|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)\Big)$
- **Computing Minimal weight feedback arc set.** $O\Big(|\mathbb{C}| (|\mathbb{R}| + |\mathbb{M}|)\Big)$
- **Generating Maximal mapping set.** $O\Big(|\mathbb{M}| (|\mathbb{M}| + |\mathbb{C}|^2)\Big)$

Hence, we can observe that the total runtime complexity of this approach turns out to be $O\Big(|\mathbb{C}| (|\mathbb{C}| + |\mathbb{M}|)(|\mathbb{C}| + |\mathbb{R}| + |\mathbb{M}|)\Big)$.

Algorithm 3.9 Computing Mapping subset $\mathbb{M}'_{\mathsf{x:y}}$ from feedback arc set $\mathbb{FAS}$

```
Require: $\mathbb{FAS}$, $\mathbb{M}_{\mathsf{x:y}}$
 1: $M := \emptyset$
 2: for all $v_i^{\mathsf{x}} \longrightarrow v_j^{\mathsf{y}} \in \mathbb{FAS}$ do
 3:   for all $c_k^{\mathsf{x}} \in v_i^{\mathsf{x}}$ and $c_l^{\mathsf{y}} \in v_j^{\mathsf{y}}$ do
 4:     $M := M \bigcup \{c_k^{\mathsf{x}} \prec c_l^{\mathsf{y}}\}$
 5:   end for
 6: end for
 7: for all $v_j^{\mathsf{y}} \longrightarrow v_i^{\mathsf{x}} \in \mathbb{FAS}$ do
 8:   for all $c_k^{\mathsf{x}} \in v_i^{\mathsf{x}}$ and $c_l^{\mathsf{y}} \in v_j^{\mathsf{y}}$ do
 9:     $M := M \bigcup \{c_k^{\mathsf{x}} \succ c_l^{\mathsf{y}}\}$
10:   end for
11: end for
12: for all $v_i^{\mathsf{x}} \longleftrightarrow v_j^{\mathsf{y}} \in \mathbb{FAS}$ do
13:   for all $c_k^{\mathsf{x}} \in v_i^{\mathsf{x}}$ and $c_l^{\mathsf{y}} \in v_j^{\mathsf{y}}$ do
14:     $M := M \bigcup \{c_k^{\mathsf{x}} \equiv c_l^{\mathsf{y}}\}$
15:   end for
16: end for
17: $\mathbb{M}'_{\mathsf{x:y}} := \mathbb{M}_{\mathsf{x:y}} \setminus M$
18: return $\mathbb{M}'_{\mathsf{x:y}}$
```

3.6.8 Correctness

In this approach our objective is simple. We are trying to find out all those mapping edges in the mapping graph representation that are participating in cycles. As we proved in Theorem 3.2, there must be at least two mapping edges in each cycle in the mapping graph. Hence, if we remove those mapping edges, then there will be no cycles in the mapping graph. The mapping relationships corresponding to the mapping edges are the one that can cause inconsistency in the combined ontology. Hence, we remove the mapping relationships corresponding to these mapping edges from the original mapping set. The remaining subset is correct and does not cause any inconsistency in the combined ontology as the set that has been removed is corresponding to a feedback arc set that has been computed using a already established correct algorithm [Demetrescu and Finocchi, 2003]. Hence, the correctness of our algorithm follows from there. Moreover, since the set being removed is the minimal set, the set of remaining mappings is a maximal subset of the input mapping set.

3.7 Discussion

In this chapter we discussed three different approaches to find the solution in polynomial time. Now, all of these approaches have some benefit over the other, hence, to compute our final heuristic solution, we will compute maximal mapping subset using all the three algorithm and then the subset with the maximum cardinality out of those three will be our solution.

We can observe that the simple naïve algorithm is very simple to implement, however, there is no guarantee of the approximation. It may be useful if the weights of mapping relationships are provided such that the information helps in maximizing the solution. Moreover, it may be useful if user has a good way to prioritize the mapping relationships.

The algorithm for biased approach is also very simple to implement. Even though it may seem that it is better than the simple naïve algorithm, it is not necessarily the case. The reason is that when we add all the mapping relationships of $\prec$ or $\succ$ type, it is possible that one of those mappings is causing conflicts with all the other mappings that are not yet added. Thus, none of these mappings would be added to the solution. However, since the total weight of these mappings may be more than those that were already added to subset, and it may be the case that they may all get added to the solution set while computing using Algorithm 3.4. Hence, this is a case when the naïve approach may give a solution set of of higher weight.

Both of these approaches are greedy in nature since both try to add as many mappings as possible in the subset. Neither of them consider what mappings are being added to the subset or what mappings are being left out except for whether they cause inconsistency or not. The graph-based approach tries to take this into consideration. It heuristically tries to identify the mappings that are actually causing more inconsistencies with other mappings and then removes them before removing the mappings that are causing less inconsistencies.

There is another subtle advantage of the graph-based approach. We know that our problem and solution builds over the mappings that were either generated using some other tool or were user-specified. Now, a lot of times very few of these mappings may cause inconsistency. Since the graph-based approach starts with all the mappings and removes a mapping at a time, occasionally, it will need to check for consistency fewer number of times than the other two approaches that try to add a mapping at a time and check for consistency after each addition.

CHAPTER 4. RESULTS AND EVALUATION

In this chapter we discuss how we evaluated our algorithms. First, we will discuss about the implementation details. Then, we will compare the accuracy of our algorithms by comparing their results with the optimal results. Later, we will compare our heuristic algorithm with each other.

4.1 Implementation Details

We chose Java language to implement all the algorithms. Our algorithms were designed to work using simple data structures corresponding to the ontologies and mapping sets. Even though the algorithms themselves are independent of the language in which the input ontologies were specified; we still added an abstraction layer that can read ontologies specified in OWL and convert those to our data structures. The simple reason to choose OWL was that it is one of the widely used ontology specification language.

Following is the list of all the tools and libraries that were used to implement our solution:

- **JDK 1.6.0_12.** We decided to use Java as the language for implementing and testing our solution. More details about Java can be found at `http://java.sun.com/javase/`.

- **JGraphT 0.8.0.** JGraphT is a free open source Java graph library of several useful algorithms. We chose to use it since it supports directed graphs, weighted and labeled edges, subgraphs, and most importantly, lets the developer use any object as vertex. More information about JGraphT is available at `http://jgrapht.sourceforge.net/`. We note that JGraphT does not allow to use bi-directional edges. Hence, we had to modify the algorithms to add support for bi-directional edges.

- **OWL API 2.2.0.** OWL API is an open source Java implementation for OWL. More details are available at http://owlapi.sourceforge.net/.
- **Pellet 2.0.0 RC5** Pellet is an open source Java reasoner for OWL. We used Pellet in combination with OWL API to parse input OWL ontologies. More details about Pellet can be found at http://clarkparsia.com/pellet/.
- **Eclipse 3.4.1.** Eclipse is a free Java IDE. More details about Eclipse can be found at http://www.eclipse.org/.

4.2 Evaluating Accuracy of Heuristic Approach

In order to evaluate the accuracy of heuristic approach, we first implemented all the three algorithms, viz, Naïve Approach (Section 3.4.2), Biased Approach (Section 3.5.2), and the Graph-based Approach (Section 3.6.1). The maximum of the three results returned by them is considered the result by the heuristic approach. Further, we also implemented the brute force algorithm (Section 3.1.2). The result obtained by it is considered to be the optimum result. We compared these two results against each other.

4.2.1 Test Setup

It was a difficult choice to decide what tests to perform in order to compare the accuracy of our algorithms, because we do not know any benchmark test cases. We decided to generate random ontologies and mapping sets and then use them as input for our algorithms. Previously Wang et al. [2006] surveyed real world ontologies and presented the distribution of the ontologies with respect to the number of concepts and relationships among other details. For example, Table 4.1 summarizes the distribution of the ontologies[1] with respect to the number of concepts in those ontologies, as surveyed by them. The number of concepts and relationships in the random ontologies follows similar distribution.

The random ontologies were generated as follows. The number of concepts in each ontology was determined on the basis of the distribution shown in Table 4.1. The number of relationships

[1]Additional information was obtained by Wang et al. [2006] to obtain these results

Table 4.1: Distribution of Ontologies with respect to No. of Concepts

No. of Concepts	No. of Ontologies (%)
0 – 50	61.5%
51 – 100	16.8%
101 – 150	6.4%
151 – 200	3.9%
201 – 250	1.4%
251 – 300	1.1%
301 – 7922	8.5%

in each ontology was further determined on the basis of distribution of relationships with respect to the number of concepts in the ontology. After selecting two ontologies at random from the generated set of ontologies, we generated random mapping sets. For the sake of simplicity, we used unit weight for each mapping relationship. We generated multiple mapping sets of size less than 32 and ran the algorithms with this input mapping set. We restricted the number of mappings to 31 since the performance of the brute force drastically degraded beyond this (takes several hours for single execution). This completes our setup.

4.2.2 Results

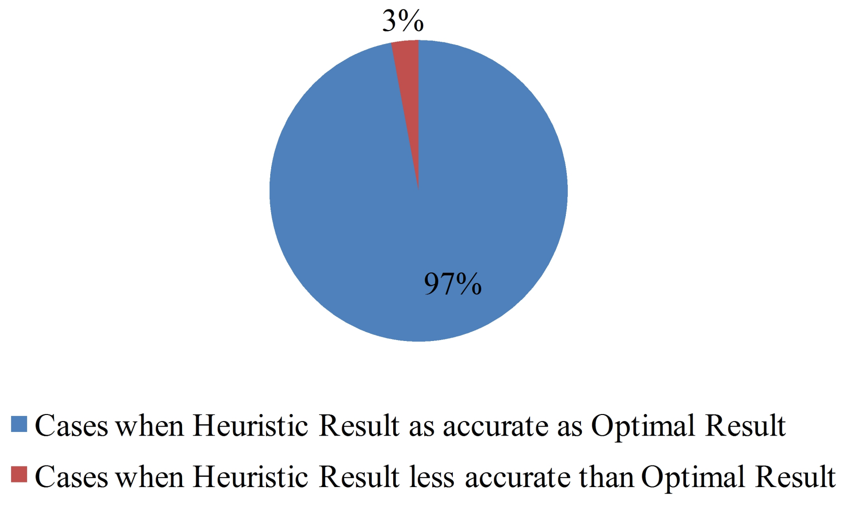


Figure 4.1: Result Distribution (Heuristic vs. Optimal)

Figure 4.1 captures the performance of the heuristic solution. Out of all the test cases that we ran (with input mapping size less than 32) 97% of the times the heurstic solution was as good as the optimal solution. Moreover, among all the cases in which the heuristic solution differed from the optimal, 94% cases differed by only 1 mapping whereas, the remaining differed by only 2 mappings with respsect to the optimal.

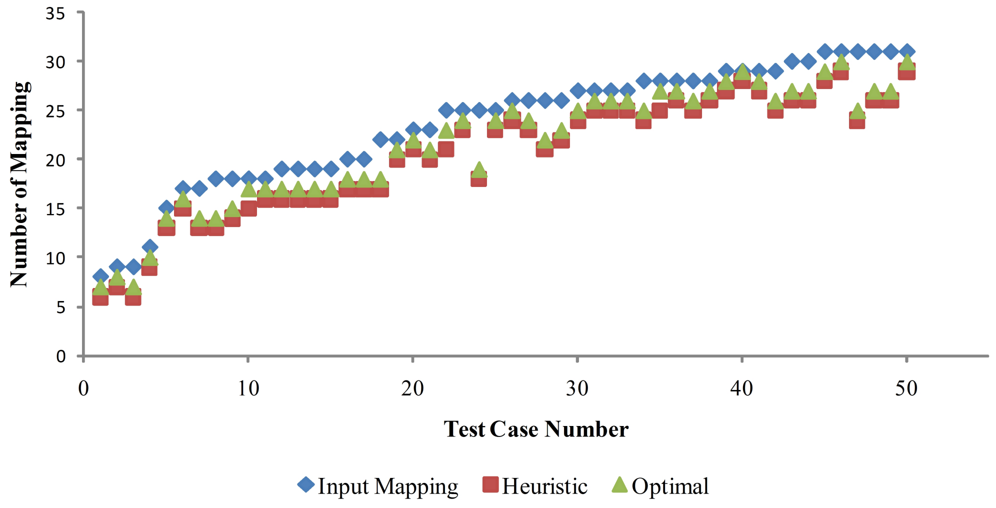


Figure 4.2: Comparison of Heuristic and Optimal results with respect to Input Size (only in case of difference)

Figure 4.2 depicts the cases in which the heuristic result was not as good as the as the optimal result. We have ordered the tests in increasing order of the number of input mappings. The graph clearly shows that for most of the cases, the difference between the two results was only 1.

4.3 Comparing the Heuristic Approaches against each other

4.3.1 Random Ontologies and Mappings

Apart from testing the accuracy of heuristic algorithms, we also compared the algorithms with each other. We followed the same test setup as discussed in the previous section. However, this time we did not restrict the number of mappings in the mapping set.

Figure 4.3 captures the performance of individual heuristic algorithms. Out of all the test

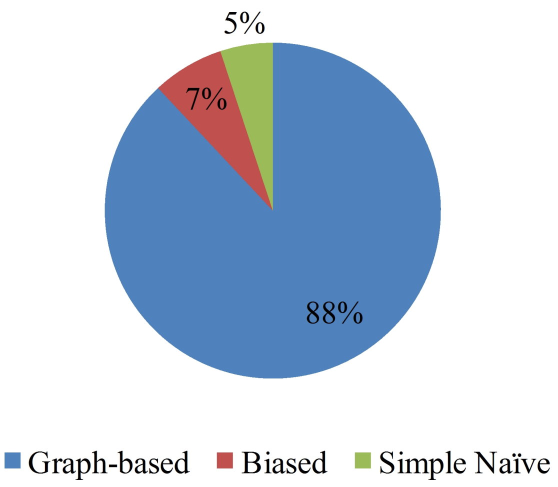


Figure 4.3: Result Distribution (Graph-based vs. Biased vs. Simple Naïve)

cases that we ran 88% of the times the graph-based heurstic algorithm outperformed the other two. This also means that almost 12% times either of the other two algorithms were better than the graph-based approach. Hence, it makes sense that we are using the maximum of the three in order to determine our final heuristic result.

Table 4.2: Improvement in the result computed by Graph-based Algorithm (input mapping size greater than 100)

Difference in	Cases when Graph-based was better than	
Number of Mappings	Simple Naïve	Biased
0 – 100	73.62%	94.60%
101 – 200	7.22%	3.29%
201 – 300	2.66%	0.74%
301 – 400	2.76%	0.29%
401 – 500	2.52%	0.34%
501 – 600	2.00%	0.29%
601 – 700	1.56%	0.05%
701 – 800	1.62%	0.10%
801 – 900	1.14%	0.10%
901 – 1000	1.00%	0.10%
1000+	3.90%	0.10%

Table 4.2 illustrates the importance of the graph-based approach. The first column lists the number of mappings that the result obtained graph-based algorithm had more than the other

algorithms. The other two columns lists the percentage of all the cases when that algorithm had as many less mappings in the solution as specified in the first column. The cases that we considered here are all the ones when the input mapping set size was more than 100. It is easy to note that for a lot of cases there is a huge difference in the result computed by these algorithms. Hence, this shows the importance of graph-based approach over the other two algorithms.

4.3.2 Real World Ontologies and Mappings

Table 4.3: Performance of the Algorithms while testing for Real World Ontologies and Mappings

Ontology	Total	Total	No. of	Avg. Execution Time (in *ms*)		
Names	Classes	Relationships	Mapping	Simple	Biased	Graph-based
animals (A, B)	17	19	9	7	2	1
people+pets (A, B)	116	147	58	100	90	2
russia (C, D)	225	243	86	264	285	2
russia (1, 2)	314	327	70	336	286	3
russia (A, B)	254	275	103	374	379	3
Sport (Soccer, Event)	570	558	148	1326	1332	8
Tourism (A, B)	814	850	190	2827	2754	124

In order to satisfy this further, we compared these three algorithms for real world ontologies and mappings such that there were very few inconsistencies. We chose to use the test ontologies and corresponding mappings given by "A Framework for Ontology Alignment and Mapping" [Ehrig and Sure, 2005]. Table 4.3 lists down the details of those ontologies and the average time taken for executing the three algorithms 10 times. Figure 4.4 compares the average execution time for these three algorithms on a logarithmic scale with respect to the increase in the size of the given ontologies and mappings.

It is clear from the table that the graph-based algorithm clearly performs much better than the other two algorithms. Since the input mappings were manually detected, they are almost accurate. Hence, as discussed in Section 3.7, the graph-based algorithm detects the solution very quickly. However, the other algorithms need to check for consistency whenever they try

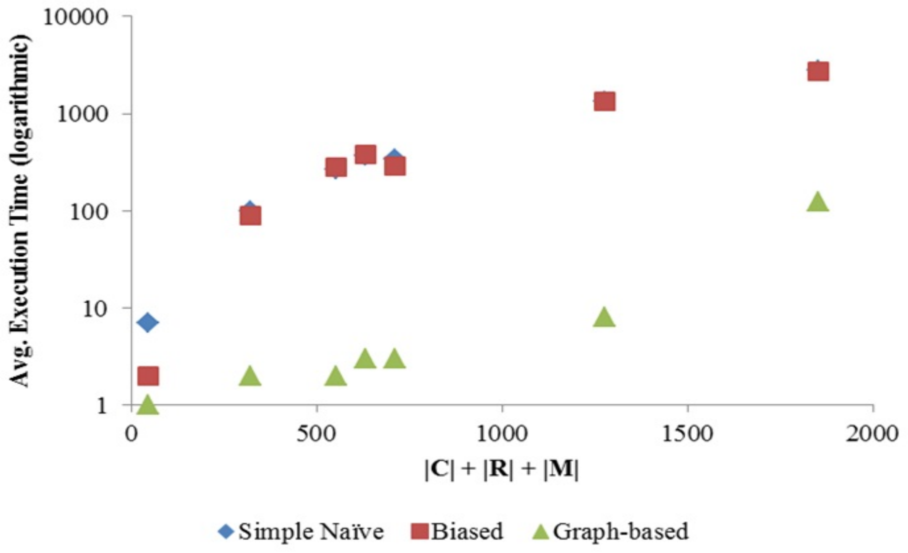


Figure 4.4: Comparison of average execution time for Simple Naïve, Biased, and Graph-based with respect to the increase in the size of the given ontologies and mappings

to add a mapping to the solution set, and hence, they end up performing poorly.

CHAPTER 5. SUMMARY AND DISCUSSION

In this chapter we will first summarize our work and then discuss some of the ways in which this work can be extended in future.

5.1 Summary

Through this work we have tried to study the problem of identifying the maximum consistent subset of mappings that may be used to combine two ontologies such that the combined ontology does not have any contradictory relationships. Therefore, we are trying to assist the process of combining two ontologies – a key step in information integration.

We present the problem of identifying the maximum consistent subset of a given mapping set that can be used to combine two consistent ontologies such that the resulting ontology also remains consistent. We formally define the optimization version of this problem within our scope. We also show that the problem is NP-hard by reducing the Minimum Feedback Arc Set in Bipartite Tournament graphs to it.

Then we discuss three different approaches to find an approximate solution for this problem. Each one had some benefit over the other. Two of these approaches, viz. simple naïve and biased approach are very simple to implement. Both of them are greedy in nature as each of them tries to increase the weight of the subset being computed. Neither of them consider what mappings are being added to the subset or what mappings are being left out except for whether they cause inconsistency or not. Therefore, each of them require several consistency checks. However, as shown by our results, for a random set of mappings there are instances when these approaches actually perform better than the third approach that we discussed.

The third and final approach that we discussed to find a heuristic solution to our problem is a graph based approach. In order to construct this solution, we model our problem as a graph problem, in particular as an edge weighted directed graph, and then use a known heuristic to identify minimal weight feedback arc set of this graph. We showed a simple construction that can be used to convert an instance of our problem to an instance of this graph problem. We also showed how we can use the known heuristic in our problem setting and how we can compute the mapping set back from the feedback arc set. As shown by our results, this solution usually performs better for the mappings that were determined manually or using some other tool.

Finally, we study the performance of all these approaches. We randomly generated several ontologies and sets of mappings between them to study the performance of our algorithms. The size of ontologies were chosen to be similar to the real world ontologies. We used a brute force solution to compute the optimal solution for this random setting and we compared our algorithms against this optimal. We found that the 97% of the times our solution was as accuate as this optimal solution. We also compared the three approaches against each other by studying them for the real world ontologies and mappings. We found that graph based approach outperforms the other two approaches in the real world setting.

5.2 Future Work

There is a lot of scope on improving and extending this work. Some of the suggested areas are as follows:

- In our work we considered the mapping relationship to be a relation between two classes, as defined in Definition 2.8. In Appendix B, we extend the problem by considering the relationships that are be a relation between two sets of classes, as defined in Definition B.1. We also discuss an approach that may be used to extend the graph based solution discussed in Section 3.6. In future, it will be worthwhile to study this approach.

- We assumed unit weight for the given mappings. It will be useful to study the performance of the algorithm for variable weight.

- Even though we tried to study the performance of our algorithms against ontologies generated at random, we had to restrict the number of mappings in the mapping set to a small value since the performance of brute force algorithm (that we used to determine optimal solution) drastically degraded beyond that. It will be worthwhile to study the performance of the algorithm for larger mapping sets.
- In this work, we expect to have a set of identified mappings between two ontologies that need to be combined. Our approach can be further extended to compare the performance of tools that are used to generate these mappings.
- It will be worthwhile to extend the prototype implementation into a more user friendly application or a tool.
- As with several heuristic solutions, it may be possible to further improvise this solution.

APPENDIX A. NOTATIONS

Table A.1: Notations

Symbol	Meaning
$\mathbb{O}$	Set of all ontologies, $\mathbb{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots\}$
$\mathcal{O}_\mathsf{x}$	x^{th} ontology where $\mathsf{x} \in \mathbb{N}$, $\mathcal{O}_\mathsf{x}\colon \langle \mathbb{C}_\mathsf{x},\ \mathbb{R}_\mathsf{x} \rangle$
$\mathbb{C}_\mathsf{x}$	Set of classes, $\mathbb{C}_\mathsf{x} = \{c_1^\mathsf{x}, c_2^\mathsf{x}, \dots\}$
c_i^x	i^{th} class
$\mathbb{R}_\mathsf{x}$	Set of relationships, $\mathbb{R}_\mathsf{x} = \{r_1^\mathsf{x}, r_2^\mathsf{x}, \dots\}$
r_p^x	p^{th} relationship, $r_p^\mathsf{x} \in \left\{c_i^\mathsf{x} \prec c_j^\mathsf{x}, c_i^\mathsf{x} \equiv c_j^\mathsf{x}\right\}$
$\mathcal{G}_{\mathcal{O}_\mathsf{x}}$	Ontology graph for $\mathcal{O}_\mathsf{x}$, $\mathcal{G}_{\mathcal{O}_\mathsf{x}}\colon \langle \mathbb{V}_{\mathcal{O}_\mathsf{x}},\ \mathbb{E}_{\mathcal{O}_\mathsf{x}} \rangle$
$\mathbb{V}_{\mathcal{O}_\mathsf{x}}$	Set of vertices, $\mathbb{V}_{\mathcal{O}_\mathsf{x}} = \{v_1^\mathsf{x}, v_2^\mathsf{x}, \dots\}$
v_i^x	i^{th} vertex (set of equivalent classes), $v_i^\mathsf{x} = \{c_{i_1}^\mathsf{x}, c_{i_2}^\mathsf{x}, \dots\}$
$\mathbb{E}_{\mathcal{O}_\mathsf{x}}$	Set of directed ontology edges, $\mathbb{E}_{\mathcal{O}_\mathsf{x}} = \{e_1^\mathsf{x}, e_2^\mathsf{x}, \dots\}$
e_p^x	p^{th} directed ontology edge, $v_i^\mathsf{x} \dashrightarrow v_j^\mathsf{x}$
$\mathbb{M}_{\mathsf{x:y}}$	Set of mapping relationships between $\mathcal{O}_\mathsf{x}$ and $\mathcal{O}_\mathsf{y}$, $\mathbb{M}_{\mathsf{x:y}} = \{r_1^{\mathsf{x:y}}, r_2^{\mathsf{x:y}}, \dots\}$
$r_p^{\mathsf{x:y}}$	p^{th} mapping relationship, $r_p^{\mathsf{x:y}} \in \left\{c_i^\mathsf{x} \prec c_j^\mathsf{y}, c_i^\mathsf{x} \succ c_j^\mathsf{y}, c_i^\mathsf{x} \equiv c_j^\mathsf{y}\right\}$
$\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$	Ontology after combining $\mathcal{O}_\mathsf{x}$, $\mathcal{O}_\mathsf{y}$, and $\mathbb{M}_{\mathsf{x:y}}$, $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}\colon \left\langle \mathbb{C}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{R}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$
$\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$	Mapping graph for $\mathcal{O}_{\mathbb{M}_{\mathsf{x:y}}}$, $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}\colon \left\langle \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}},\ \mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} \right\rangle$
$\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$	Set of vertices, $\mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}} = \left\{v_1^\mathsf{x}, v_2^\mathsf{x}, \dots, v_1^\mathsf{y}, v_2^\mathsf{y}, \dots\right\}$
$\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}$	Set of directed edges, $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}} = \left\{e_1^\mathsf{x}, e_2^\mathsf{x}, \dots, e_1^\mathsf{y}, e_2^\mathsf{y}, \dots, e_1^{\mathsf{x:y}}, e_2^{\mathsf{x:y}}, \dots\right\}$
$e_p^{\mathsf{x:y}}$	p^{th} mapping edge, $e_p^{\mathsf{x:y}} \in \left\{v_i^\mathsf{x} \longrightarrow v_j^\mathsf{y}, v_j^\mathsf{y} \longrightarrow v_i^\mathsf{x}, v_i^\mathsf{x} \longleftrightarrow v_j^\mathsf{y}\right\}$

APPENDIX B. PROBLEM EXTENSION

So far we have considered simple mappings between two ontologies. Now, here we extend our graph-based algorithm to handle other complicated relationships between two ontologies.

Complex Mapping Relationships

Earlier in Definition 2.8 in Section 2.4 we defined a mapping relationship as a relation between two classes such that each class is from different given ontologies. Now, even though that definition allows us to related two classes, it does not allow us to capture relationships that may exist between a group of classes.

Example B.1. Consider the following example ontologies:

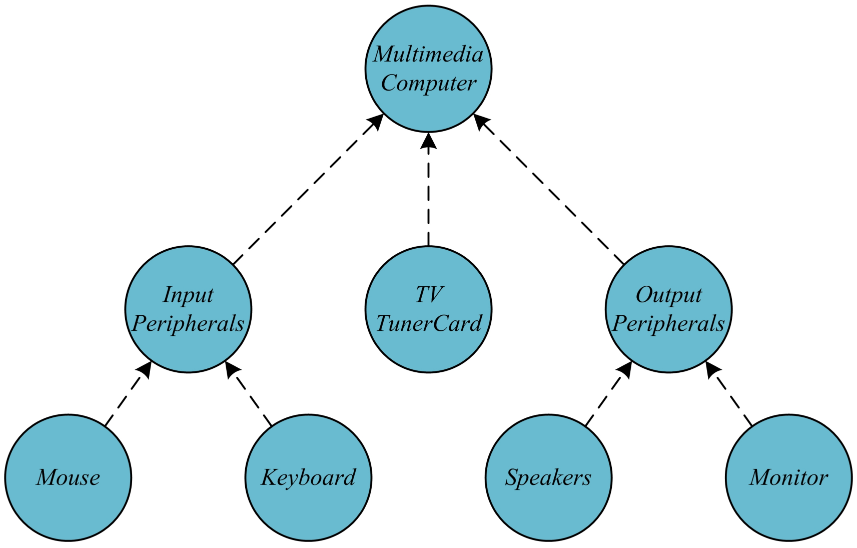
$$\mathcal{O}_1 : \left\langle \left\{ \begin{array}{l} MultipediaComputer, \\ InputPeripherals, \\ OutputPeripherals, \\ Keyboard,\ Mouse, \\ Speakers,\ Monitor, \\ TVTunerCard \end{array} \right\}, \left\{ \begin{array}{l} InputPeripherals \prec MultimediaComputer, \\ OutputPeripherals \prec MultimediaComputer, \\ Mouse \prec InputPeripherals, \\ Keyboard \prec InputPeripherals, \\ Speakers \prec OutputPeripherals, \\ Monitor \prec OutputPeripherals, \\ TVTunerCard \prec MultimediaComputer \end{array} \right\} \right\rangle$$

$$\mathcal{O}_2 : \left\langle \left\{ \begin{array}{l} EntertainmentDevices, \\ Television,\ Radio \end{array} \right\}, \left\{ \begin{array}{l} Television \prec EntertainmentDevices, \\ Radio \prec MultimediaComputer \end{array} \right\} \right\rangle$$

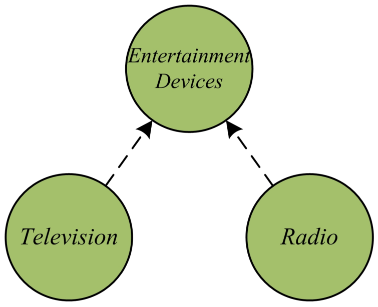
Figure B.1 shows the ontology graphs corresponding to $\mathcal{O}_1$ and $\mathcal{O}_2$ for better visualization. Now, some of the possible mappings between the two ontologies may be:

$$MultimediaComputer \text{ without } InputPeripherals \text{ is equivalent to } Television \qquad \text{(B.1)}$$

The important thing to note here is that the relationship holds only when everything that constitutes a $MultimediaComputer$ and that is not an $InputPeripherals$ in $\mathcal{O}_1$ is equivalent to a $Television$ in $\mathcal{O}_2$.



(a) $\mathcal{G}_{\mathcal{O}_1}$



(b) $\mathcal{G}_{\mathcal{O}_2}$

Figure B.1: Example for Complex Mapping Relationship (we skip the { and } symbols in vertices for simplicity)

Consider another possible mapping:

$$OutputPeripherals \text{ and } TVTunerCard \text{ together are equivalent to } Television \tag{B.2}$$

It is important to note here that both $OutputPeripherals$ and $TVTunerCard$ *together* are equivalent to $Television$. None of them is equivalent to $Television$ on its own.

This example shows that there is a need for us to enhance the definition of mapping relationship in a way which can capture these scenarios.

Definition B.1 (Mapping Relationship)**.** Given ontologies $\mathcal{O}_\mathsf{x} \colon \langle \mathbb{C}_\mathsf{x},\ \mathbb{R}_\mathsf{x} \rangle$ and $\mathcal{O}_\mathsf{y} \colon \langle \mathbb{C}_\mathsf{y},\ \mathbb{R}_\mathsf{y} \rangle$, a *mapping relationship* $r^{\mathsf{x}:\mathsf{y}}$, is a relation $\mathbb{C}'_\mathsf{x} \mathcal{R} \mathbb{C}'_\mathsf{y}$ between any two subset of classes $\mathbb{C}'_\mathsf{x} \subseteq \mathbb{C}_\mathsf{x}$, $\mathbb{C}'_\mathsf{y} \subseteq \mathbb{C}_\mathsf{y}$ where $\mathcal{R}$ is either of the following:

$\prec$ **Subclass relation.** $\mathbb{C}'_\mathsf{x} \prec \mathbb{C}'_\mathsf{y}$ represents that all classes in the set $\mathbb{C}'_\mathsf{x}$ together are subclass of all classes in set $\mathbb{C}'_\mathsf{y}$, that is, $\forall c^\mathsf{x}_i \in \mathbb{C}'_\mathsf{x},\ c^\mathsf{y}_j \in \mathbb{C}'_\mathsf{y} \colon c^\mathsf{x}_i \prec c^\mathsf{y}_j$

$\succ$ **Super class relation.** $\mathbb{C}'_\mathsf{x} \succ \mathbb{C}'_\mathsf{y}$ represents that all classes in the set $\mathbb{C}'_\mathsf{x}$ together are super class of all classes in set $\mathbb{C}'_\mathsf{y}$, this is, $\forall c^\mathsf{x}_i \in \mathbb{C}'_\mathsf{x},\ c^\mathsf{y}_j \in \mathbb{C}'_\mathsf{y} \colon c^\mathsf{x}_i \succ c^\mathsf{y}_j$

$\equiv$ **Equivalence relation.** $\mathbb{C}'_\mathsf{x} \equiv \mathbb{C}'_\mathsf{y}$ represents that all classes in the set $\mathbb{C}'_\mathsf{x}$ together are equivalent to all classes in set $\mathbb{C}'_\mathsf{y}$, this is, $\forall c^\mathsf{x}_i \in \mathbb{C}'_\mathsf{x},\ c^\mathsf{y}_j \in \mathbb{C}'_\mathsf{y} \colon c^\mathsf{x}_i \equiv c^\mathsf{y}_j$

Note. We note that the properties of the relations defined in Section 2.1 still hold.

Remark. Now, this is quite broad definition of mapping relationship and this way, along with some *additional symbols*, we can capture the scenarios that we discussed earlier. For e.g. Equation (B.2) can be represented as

$$\{OutputPeripherals, TVTunercard\} \equiv \{Television\}$$

and Equation (B.1) can be represented as

$$MultimediaComputer \setminus InputPeripherals \equiv \{Television\}$$

where $a \setminus b$ means that all subclasses of a except for b.

We can extend this for other cases like, for example, when we want to relate some of the super classes of a class, rather than the subclasses. We note that all this can be achieved with the help of some additional symbols and for the remaining discussion, we will not consider how this is being done.

Simple Naïve Approach and Biased Approach

The simple naïve approach (Section 3.4) and the biased approach (Section 3.5) presented earlier would still work fine. We can view a complex mapping relationship defined in Definition B.1 as a group of simple mapping relationships as defined Definition 2.8. Now, at a time rather than adding one simple mapping and checking for consistency, in order to decide whether that mapping will be retained or not; we will add this group of simple mappings and check for consistency. If the combined ontology is consistent, then we retain this complex mapping relationship, otherwise, we ignore it.

Graph-based Approach

Here we will present how an addition of $|\mathbb{M}|$ special vertices to our mapping graph will help us in solving the problem containing mapping relationships with little change to our graph-based heuristic approach.

Earlier in Section 3.3 we showed that we will connect two vertices of different ontology graphs with a special mapping edge that corresponds to a mapping relationship. Now, we modify that construction. Now, rather then joining the vertices of different ontology graphs directly with edges, we will introduce some intermediate vertices, called *corridor vertices* with some special properties. Here is how we will represent our mapping graph now.

Given any two ontologies $\mathcal{O}_\mathsf{x}\colon \langle \mathbb{C}_\mathsf{x},\ \mathbb{R}_\mathsf{x} \rangle$ and $\mathcal{O}_\mathsf{y}\colon \langle \mathbb{C}_\mathsf{y},\ \mathbb{R}_\mathsf{y} \rangle$ and a mapping set $\mathbb{M}_\mathsf{x:y}$ (containing complex mapping relationships as defined in Definition B.1), $\mathcal{G}_{\mathbb{M}_\mathsf{x:y}}\colon \left\langle \mathbb{V}_{\mathbb{M}_\mathsf{x:y}},\ \mathbb{E}_{\mathbb{M}_\mathsf{x:y}} \right\rangle$ represent the mapping graph generated by combining $\mathcal{G}_{\mathcal{O}_\mathsf{x}}$ and $\mathcal{G}_{\mathcal{O}_\mathsf{y}}$ using $\mathbb{M}_\mathsf{x:y}$ where,

$\mathbb{V}_{\mathbb{M}_\mathsf{x:y}}$ is a finite set of vertices, such that $\mathbb{V}_{\mathbb{M}_\mathsf{x:y}} = \mathbb{V}_{\mathcal{O}_\mathsf{x}} \bigcup \mathbb{V}_{\mathcal{O}_\mathsf{y}} \bigcup \mathbb{V}$

$\mathbb{V}$ is a finite set of *corridor vertices*, that is $\mathbb{V} = \left\{\partial_1^\mathsf{x}, \partial_1^\mathsf{y}, \partial_2^\mathsf{x}, \partial_2^\mathsf{y}, \ldots\right\}$ such that $|\mathbb{V}| = 2\,|\mathbb{M}_\mathsf{x:y}|$ and $\forall \partial_p^\mathsf{x} \in \mathbb{V}\colon \exists \partial_p^\mathsf{y} \in \mathbb{V}$ such that ∂_p^x and ∂_p^y are connected with a mapping edge

$\mathbb{E}_{\mathbb{M}_\mathsf{x:y}}$ is a finite set of labeled directed edges such that $\mathbb{E}_{\mathbb{M}_\mathsf{x:y}} = \mathbb{E}_{\mathcal{O}_\mathsf{x}} \bigcup \mathbb{E}_{\mathcal{O}_\mathsf{y}} \bigcup \mathbb{E}$

$\mathbb{E}$ is a set of *mapping edges* generated using the mapping relationships such that each edge is either of the following:

- $v_i^{\mathsf{x}} \longrightarrow \partial_p^{\mathsf{x}}$
- $\partial_p^{\mathsf{x}} \longrightarrow v_i^{\mathsf{x}}$
- $v_i^{\mathsf{x}} \longleftrightarrow \partial_p^{\mathsf{x}}$
- $\partial_p^{\mathsf{x}} \longrightarrow \partial_p^{\mathsf{y}}$
- $\partial_p^{\mathsf{y}} \longrightarrow \partial_p^{\mathsf{x}}$
- $\partial_p^{\mathsf{x}} \longleftrightarrow \partial_p^{\mathsf{y}}$
- $\partial_p^{\mathsf{y}} \longrightarrow v_j^{\mathsf{y}}$
- $v_j^{\mathsf{y}} \longrightarrow \partial_p^{\mathsf{y}}$
- $\partial_p^{\mathsf{y}} \longleftrightarrow v_j^{\mathsf{y}}$

such that

$$\begin{aligned}(\mathbb{C}'_{\mathsf{x}} \prec \mathbb{C}'_{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}) \Leftrightarrow & \left(\exists \partial_p^{\mathsf{x}} \longrightarrow \partial_p^{\mathsf{y}} \in \mathbb{E}\right) \\ & \wedge \left(\forall c_i^{\mathsf{x}} \in \mathbb{C}'_{\mathsf{x}} \ \text{vertex}\left(c_i^{\mathsf{x}}\right) \longrightarrow \partial_p^{\mathsf{x}} \in \mathbb{E}\right) \\ & \wedge \left(\forall c_j^{\mathsf{y}} \in \mathbb{C}'_{\mathsf{y}} \ \partial_p^{\mathsf{y}} \longrightarrow \text{vertex}\left(c_j^{\mathsf{y}}\right) \in \mathbb{E}\right)\end{aligned}$$

$$\begin{aligned}(\mathbb{C}'_{\mathsf{x}} \succ \mathbb{C}'_{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}) \Leftrightarrow & \left(\exists \partial_p^{\mathsf{y}} \longrightarrow \partial_p^{\mathsf{x}} \in \mathbb{E}\right) \\ & \wedge \left(\forall c_i^{\mathsf{x}} \in \mathbb{C}'_{\mathsf{x}} \ \partial_p^{\mathsf{x}} \longrightarrow \text{vertex}\left(c_i^{\mathsf{x}}\right) \in \mathbb{E}\right) \\ & \wedge \left(\forall c_j^{\mathsf{y}} \in \mathbb{C}'_{\mathsf{y}} \ \text{vertex}\left(c_j^{\mathsf{y}}\right) \longrightarrow \partial_p^{\mathsf{y}} \in \mathbb{E}\right)\end{aligned}$$

$$\begin{aligned}(\mathbb{C}'_{\mathsf{x}} \equiv \mathbb{C}'_{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}) \Leftrightarrow & \left(\exists \partial_p^{\mathsf{x}} \longleftrightarrow \partial_p^{\mathsf{y}} \in \mathbb{E}\right) \\ & \wedge \left(\forall c_i^{\mathsf{x}} \in \mathbb{C}'_{\mathsf{x}} \ \text{vertex}\left(c_i^{\mathsf{x}}\right) \longleftrightarrow \partial_p^{\mathsf{x}} \in \mathbb{E}\right) \\ & \wedge \left(\forall c_j^{\mathsf{y}} \in \mathbb{C}'_{\mathsf{y}} \ \partial_p^{\mathsf{y}} \longleftrightarrow \text{vertex}\left(c_j^{\mathsf{y}}\right) \in \mathbb{E}\right)\end{aligned}$$

Example B.2. Consider the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ from Example B.1. Figure B.2 shows mapping graph for $\mathbb{M}_{1:2} = \{\{OutputPeripherals, TVTunercard\} \equiv \{Television\}\}$

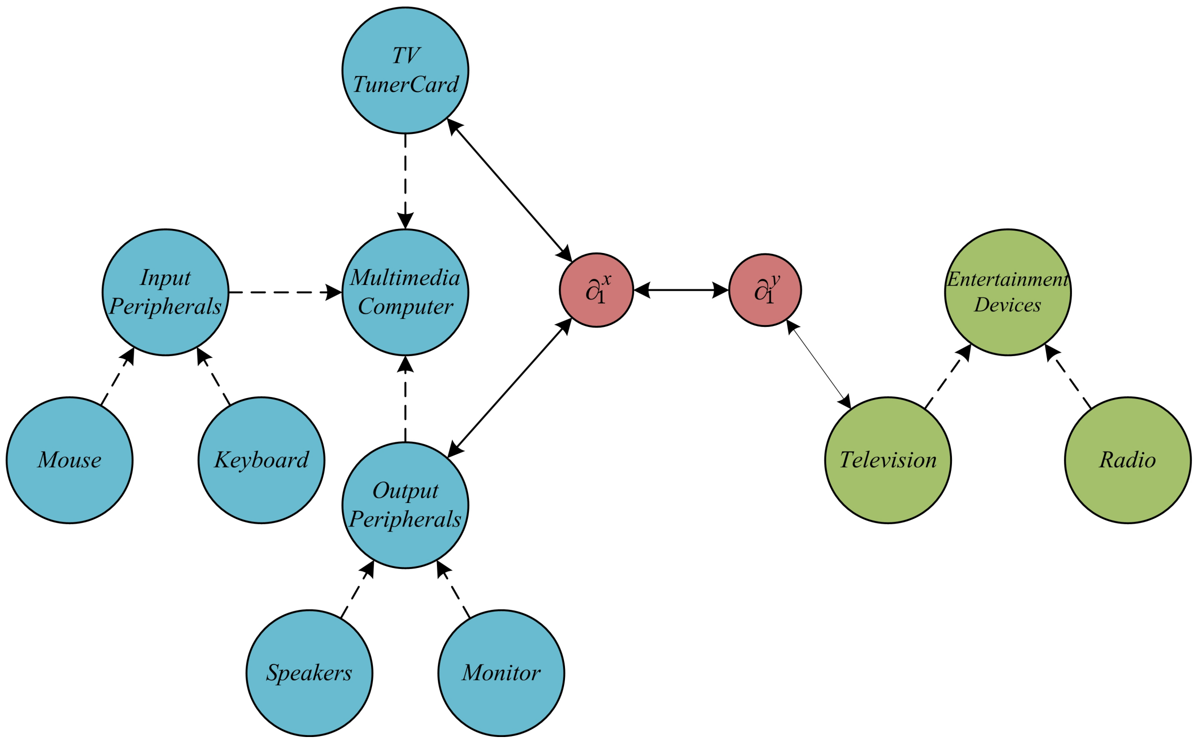


Figure B.2: Mapping graph $\mathcal{G}_{\mathbb{M}_{1:2}}$ for Example B.2

Property B.1. *For each complex mapping relationship* $r_p^{\mathsf{x:y}} = \mathbb{C}'_{\mathsf{x}} \mathcal{R} \mathbb{C}'_{\mathsf{y}} \in \mathbb{M}_{\mathsf{x:y}}$ *there are:*

- 2 *vertices, viz.* $\partial_p^{\mathsf{x}}, \partial_p^{\mathsf{y}} \in \mathbb{V}_{\mathbb{M}_{\mathsf{x:y}}}$
- $\left(|\mathbb{C}'_{\mathsf{x}}| + |\mathbb{C}'_{\mathsf{y}}| + 1\right)$ *mapping edges in* $\mathbb{E}_{\mathbb{M}_{\mathsf{x:y}}}$

Property B.2 (Path restriction on Corridor Vertices)**.** *Given a mapping graph,* $\mathcal{G}_{\mathbb{M}_{\mathsf{x:y}}}$*, the corridor vertices* ∂_p^{x} *or* ∂_p^{y} *participate in any path only if both of them participate in the path such that either of them precedes other depending on the direction of edge with which they are connected.*

This property implies that any path in which a corridor vertex participate, the mapping edge with which this corridor vertex is connected to the other corridor vertex will also participate. This restriction is very useful us, since it also means that any cycle in which a corridor vertex participates, even the mapping edge with which it is connected to the other corridor vertex will participate. Moreover, if this particular mapping edges between two corridor vertices is removed along with the corridor vertices themselves, then all the other mapping edges with which the corridor vertices are connected to other vertices will also get removed. In effect, removing one complex mapping relationship will mean none of the classes involved in the relationships will be anymore connected (they may be still connected due to some other mapping relationship).

Now, the remaining part of the solution is straight-forward. When constructing the edge weighted directed graph, all the edges other than those connected by two corridor vertices are given a weight of ∞ so that they do not get removed while computing the minimal feedback arc set. Therefore, only edges that will constitute the feedback arc set will be the mapping edges between two corridor vertices. Moreover, whenever such an edge is removed, the corridor vertices connected by that mapping edge will not participate in any more cycles, since the conditions discussed above won't meet. Once the feedback arc has been computed, we can compute the mapping subset directly by removing all the mappings corresponding to them from the original set of mappings.

BIBLIOGRAPHY

Alon, N. (2006). Ranking tournaments. *SIAM J. Discret. Math.*, 20(1):137–142.

Antoniou, G. and Harmelen, F. v. (2008). *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems).* The MIT Press.

Bonatti, P., Deng, Y., and Subrahmanian, V. (2003). An ontology-extended relational algebra. *Information Reuse and Integration, 2003. IRI 2003. IEEE International Conference on*, pages 192–199.

Charbit, P., Thomassé, S., and Yeo, A. (2007). The minimum feedback arc set problem is np-hard for tournaments. *Comb. Probab. Comput.*, 16(1):1–4.

Choi, N., Song, I.-Y., and Han, H. (2006). A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41.

Colomb, R. M. (2007). *Ontology and the Semantic Web.* IOS Press.

Demetrescu, C. and Finocchi, I. (2003). Combinatorial algorithms for feedback problems in directed graphs. *Inf. Process. Lett*, 86:2003.

Ehrig, M. and Sure, Y. (2005). Foam - a framework for ontology alignment and mapping. Demo at ISWC2005.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching.* Springer Berlin Heidelberg New York.

Falconer, S. M. and Storey, M.-A. D. (2007). A cognitive support framework for ontology mapping. In *ISWC/ASWC*, pages 114–127.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.

Guo, J., Hüffner, F., and Moser, H. (2007). Feedback arc set in bipartite tournaments is np-complete. *Inf. Process. Lett.*, 102(2-3):62–65.

Johnson, D. B. (1975). Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84.

Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1):1–31.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Shvaiko, P. and Euzenat, J. (2008). Ten challenges for ontology matching. pages 1164–1182.

Wang, T. D., Parsia, B., and Hendler, J. (2006). A survey of the web ontology landscape. In *International Semantic Web Conference*, pages 682–694.