Developing Materials Ontology On The Phase Diagram Knowledge Graph

A THESIS

Submitted by

Tapish Devendra Garg

for the award of the degree

of

BACHELORS OF TECHNOLOGY

DEPARTMENT OF METALLURGICAL AND MATERIALS ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

CHENNAI-600036

June 2021

THESIS CERTIFICATE

This is to certify that the thesis entitled **"Developing Material Ontology on the phase diagram knowledge graph"** submitted by **Tapish Devendra Garg** to the Indian Institute of Technology, Madras for the award of the degree of **Bachelors of Technology** is a bonafiderecord of research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award ofany degree or diploma.

Dr. Gandham Phanikumar
Professor
Department of Metallurgical and Materials Engineering
Indian Institute of Technology Madras
Chennai – 600 036.

Place: Chennai

Date: June 2021

ACKNOWLEDGEMENTS

It is with immense pleasure and satisfaction that I take this opportunity to express my wholehearted gratitude in a few words and respect to all those who have helped and supported me on my way to this point.

I would like to extend my special thanks of gratitude to my B.Tech Project guide ***Dr. Gandham Phanikumar*** for the golden opportunity to work on this wonderful project. His valuable guidance and support have been constantly encouraging and has helped me achieve a lot academically and well as professionally.

I also want to thank V.S. Hariharan for his invaluable assistance and helpful discussion on topics related to my project work. I am greatly obliged to all the faculty and staff members of the Department of Metallurgical and Materials Engineering who have directly or indirectly helped me in my coursework.

ABSTRACT

Keywords: Ontology, knowledge graph, phase diagram, alloys, SPARQL, pymatgen, owlready2, Protégé

For materials science and engineering, informatics plays a significant role in innovation. New material discovery, materials selection and failure analysis all rely primarily on a wide range of materials information and knowledge. We have a wealth of information resources, including books, research papers, the internet, digitized libraries and many other databases. However, the majority of materials data is dispersed among multiple heterogeneous sources with no proper links. These resources fail to collect the crucial metadata that represents information like the relationship between materials data. Therefore it is difficult for an individual to get desired information effectively.

It is vital to store materials data in a way that makes it findable and reusable in order to optimize the use of data created by various researchers and organizations. Complex and advanced inquiries (which contains more than one concept) would be possible if this material's data is collected and organized alongside the original data coming from existing resources. To address this issue, we have built these relationships between different concepts. This project uses the ontology language of the semantic web, OWL, which enables the definition of the flexible and detailed structure of materials information, SPARQL, to query the data stored in RDF format.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....3

ABSTRACT.....4

TABLE OF CONTENTS.....5

LIST OF TABLES.....7

LIST OF FIGURES.....8

Chapter 1: Introduction.....10

1.1 Introduction.....10

1.2 Objective of the Work.....11

CHAPTER 2: Literature Review.....12

2.1 Related Work.....12

2.2 Background.....12

CHAPTER 3: Methodology.....18

3.1 Creating Elements class.....18

3.2 Creating Alloys Class.....21

3.3 Creating Phase Diagram Features Class.....21

3.4 Creating Phase Properties Class.....24

3.5 Hierarchy and Querying data.....26

CHAPTER 4: Results and Discussion.....28

CHAPTER 5: Summary and Conclusions.....33

5.1 Prospects Generated by this Work: *33*

5.2 Recommendations for Future Work: *34*

Appendix *35*

References *44*

LIST OF TABLES

Table 1 – Example Query…….15
Table 2 – Data properties of Elements class with range and label……19
Table 3 – Data Properties of Phases class with range and label……23
Table 4 – All object properties in AlloyOnto with domain, range and inverse property……..27
Table 5 – Query 1……..29
Table 6 – Query 2……..29
Table 7 – Query 3……..30
Table 8 – Query 4……..31
Table 9 – Query 5……..32

LIST OF FIGURES

Fig 1 – Class Book has relation with class Author through has Author object property……13

Fig 2 – To killing a MockingBird is an individual of class Book and Harper Lee is an individual of class Author. So the above graph means that To Killing a MockingBird has author Harpar Lee……14

Fig 3 – Class BinaryAlloys is a subclass of Alloys……...21

Fig 4 – FeC (Iron-Carbon) is an individual of class BinaryAlloys and Fe (Iron) is an individual of class Elements. So the above graph means that FeC binary alloy contains Fe element……21

Fig 5 – FeC (Iron-Carbon) is an individual of class BinaryAlloys and Fe (Iron) is an individual of class Elements. So the above graph means that Fe element is contained in FeC……..21

Fig 6 – Invariant_Reaction, Phase and Reaction are the three subclasses of class Phase_Diagram_Features…….22

Fig 7 – Cobalt-Hafnium is an individual of BinaryAlloys and Eutectic is an individual of Invariant_Reaction. So the above graph means Cobalt-Hafnium has Eutectic reaction……22

Fig 8 – Cobalt-Hafnium is an individual of BinaryAlloys and Eutectic is an individual of Invariant_Reaction. So the above graph means Eutectic reaction is present in Cobalt-Hafnium binary alloy……...22

Fig 9 – Cobalt-Hafnium is an individual of BinaryAlloys and Hf2Co is an individual of Phase. So the above graph means Cobalt-Hafnium binary alloy has Hf2Co phase…….23

Fig 10 – Cobalt-Hafnium is an individual of BinaryAlloys and Hf2Co is an individual of Phase. So the above graph means Hf2Co phase is present in Cobalt-Hafnium binary alloys…….23

Fig 11 – GeYb_R3 is an individual of Reaction and Eutectic is an individual of Invariant_Reaction. So the above graph means GeYb_R3 reaction is a type of Eutectic reaction where GeYb_R3 represents “L2 + Yb3Ge4.3 rt = Yb10Ge11 reaction……24

Fig 12 – GeYb_R3 is an individual of Reaction and Ytterbium-Germanium is an individual of BinaryAlloy. So the above graph means GeYb_R3 reaction is a contained in Ytterbium-Germanium……...24

Fig 13 – GeYb_R3 is an individual of Reaction and Ytterbium-Germanium is an individual of BinaryAlloy. So the graph means Ytterbium-Germanium contains GeYb_R3 reaction……..24

Fig 14 – Pearson_Symbol and Space_Group are two subclasses of class Phase_Properties….24

Fig 15 – HgPt is an individual of Phase and tP2 is an individual of Pearson_Symbol. So the above graph means HgPt phase has tP2 pearson symbol…….25

Fig 16 – HgPt is an individual of Phase and tP2 is an individual of Pearson_Symbol. So the above graph means tP2 pearson symbol present in HgPt phase……25

Fig 17 – HgPt is an individual of Phase and P4/mmm is an individual of Space_Group. So the above graph means HgPt phase has P4/mmm space group……...25

Fig 18 – HgPt is an individual of Phase and P4/mmm is an individual of Space_Group. So the above graph means P4/mmm space group is present in HgPt phase…….26

Fig 19 – This graph is the complete hierarchy of our ontology which shows the relation between the classes. At the bottom right corner, meaning of each shape is mentioned……..26

Fig 20 – Graph representation of SPARQL query 1…….28

Fig 21 – Graph representation of SPARQL query 2……..29

Fig 22 – Graph representation of SPARQL query 3……..30

Fig 23 – Graph representation of SPARQL query 4…….31

Fig 24 – Graph representation of SPARQL query 5…….32

Chapter 1: Introduction

1.1 Introduction

With the advancement of technology, the range of alloys available is continually expanding in terms of both type and quantity. In the last few decades, many research institutes and organizations have collected and generated a massive amount of data which gives huge domain knowledge which is essential to materials scientists. This information on alloys really comes in handy in various applications like material selection, failure analysis and fault diagnosis [1]. There are some sources that are already present to get information on alloys like ASM Database [2] and Smithells metals reference book [3]. This database helps researchers to explore alloys properties data and phases.

But it is pretty challenging to use these existing alloy information sources to researcher benefits. The information that is present in these sources is not able to catch relations between alloys information. The classification of alloys on a particular aspect or property is quite difficult as the elements knowledge is mostly hidden in many of the materials data sources. So to link this information together, we need to need to integrate them semantically and represent them in a unified way. After representing them properly, it can be easy to discover relations between different concepts like alloys, elements, phase diagrams and their properties. One can suggest that we can do this using relational database such as MySQL by linking the tables of alloys and elements. But this type of relationship will not make any semantic sense and we need to define the proper database schema, which may not be flexible.

One solution that we have come up with is to use semantic web technologies like ontologies, RDF and SPARQL [4]. Ontologies can help us to enrich metadata by including several features of this language. For instance, suppose we have Iron-Carbon as an alloy. This alloy is also known by name FeC or CFe or Iron-Carbon, but they represent a single thing only. So this issue

will be considered in the synonym category, which can be solved using ontology. Ontology also possesses the ability to represent complex heterogeneous information.

1.2 Objective of the Work

1. To represent the knowledge about alloys and elements in ontology and derive relations between them by collecting information from existing resources of materials knowledge bases.
2. To derive a method for expressing implicit alloy selection knowledge through instance and rules.
3. To retrieve the desired results for a complex and advanced alloy based query.

We have tried to achieve all the above objectives in logical and efficient way with help of effective open source tools. Hence, the work presented here will establish the groundwork for developing an open, shared, and scalable knowledge framework for alloy selection.

CHAPTER 2: Literature Review

2.1 Related Work

Semantic web application is becoming quite popular and it addresses many issues. Engineering fields like biomedical informatics, geographical informatics [5], chemical informatics [6] and processing informatics [7] are using them for a while now. On the other hand, the application of ontologies and materials knowledge is quite new and hasn't been implemented that much. Even so there are some applications of ontologies in the materials domain that exist, such as Plinius [8] and Ashino [9] materials ontology. Plinius is an ontology which that handles knowledge about ceramic materials that helps in conceptualizing the ceramic properties. Ashino is an ontology that is designed for generic materials science and mainly focuses on creep data analysis processes and its properties.

2.2 Background

We would like to explain some concepts and terminologies in this section so that it would easy to follow through this report.

2.2.1. Ontology and OWL

An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them which is used for knowledge representation. Knowledge Representation is a branch of artificial intelligence that with the representation of human knowledge in digital format. Knowledge representation is a model which helps to capture reality needed for reasoning and inferencing between knowledge and data. So to describe knowledge representation, ontology is used, which defines concepts and rules required for real world problems as in our case Alloys. In simple words, ontology is generalized models that will store concepts and properties that are shared between individuals.

There are several ontology modeling languages that can describe ontology in which mathematical axioms are used to define rules and inferences. Here we have used web ontology language (OWL) which will help us to verify the consistency of our materials knowledge. OWL constructs ontology in the following way:

Classes - distinct type of entities exist in the domain

Examples - Materials, Elements, Alloys

Individuals - instances of classes

Examples - FeC, Iron

Properties - relationships between classes

Examples - containsElement, hasTemperature

The data inside the ontology is stored as a resource description framework (RDF). The data is represented as triples format like (subject, predicate, object). The whole knowledge is then connected as a graph which comprised of nodes and edges. RDF knowledge is represented as a directed graph. Here the node contains physical entities like concepts and classes while the directed edges show the relationship between the connected nodes or entities.

To understand it better, let us take an example. Suppose we need to represent the data of a library contains information regarding books and authors. In this case, we will Author and Book as classes which will have relation “has author”.

The above information can be represented in RDF as following:

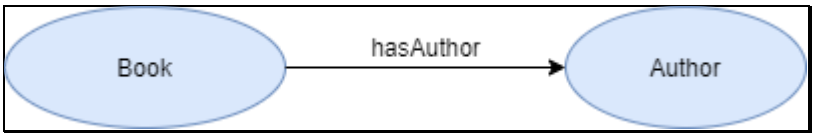

Fig 1 – Class Book has relation with class Author through has Author object property

2.2.2. Knowledge Graph

The knowledge graph is made up of interconnected descriptions of items such as objects, events, and concepts. Knowledge graphs provide a framework for data integration, unification, analytics, and sharing by putting data in context through linkage and semantic information. After defining ontology as a framework, we can add the individuals and real data in classes and concepts which are defined in the ontology.

Knowledge Graph = Ontology + Data

So suppose we have “To kill a Mockingbird” is a instance of Book class and “Harper Lee” is an instance of Author class. For the (Book → hasAuthor → Author) relationship we will represent this data as shown in the figure:

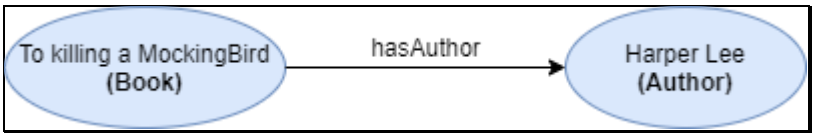

Fig 2 – To killing a MockingBird is an individual of class Book and Harper Lee is an individual of class Author. So the above graph means that To Killing a MockingBird has author Harpar Lee

2.2.3. Semantic Retrieval using SPARQL

SPARQL can combine ontological concepts and attributes with instances, production rules, and SQL instructions in a powerful way. As a result, the query statement can be used to create an interface between the alloy selection and the knowledge base. Retrieval of results is divided into two processes. First, the researcher or user at the user interface level converts his question into a SPARQL query statement and submits it to the reasoner layer. After submitting the query, the reasoner finds and extracts the required information from the knowledge base and then returns the query result to the allot selector.

Continuing to above example, let us understand the information retrieval process:

English - "Who is the author of *To kill a Mockingbird book*?"

Sparql –

```
SELECT ?author
WHERE{
        a:To_kill_a_mockingbird b:has_author ?author .
}
```

Result –

Table 1 – Example Query

?author
Harper Lee

2.2.4. Phase Diagram

A phase diagram depicts all of the equilibrium phases as a function of temperature, pressure, and composition in a graphical format. When one of the system's parameters is changed, a phase diagram shows what phases exist at equilibrium and what Phase changes we might predict. Real materials are almost always combinations of multiple elements rather than pure

substances, and composition is a variable in addition to T and P.

In solid physics and materials research, phase diagrams are essential tools. Many alloys and several oxides have well-studied and well-known phase diagrams. However, there is a scarcity of information about Nano alloy phase diagrams. This is due to the fact that this field of study is still relatively new, and creating new phase diagrams is challenging. Metallurgists use phase diagrams to choose alloys with specific compositions and to plan and regulate heat treatment methods that generate certain qualities. They're also utilized to diagnose quality issues.

2.2.4. Tools

We have used some tools that helped us to create our alloys ontology (AlloyOnto):

a) Protégé: Protégé [10] [11] is a free open source editor from Stanford University which is used for making ontology. It provides an interactive graphical user interface to define ontology. It consists of various reasoners (we have used Pellet) to check consistency and infer new information on the basis of an ontology. Continuing our above example, we have defined the relationship:

(Book → has_author → Author)

And we have also defined has_book is the inverse property of has_author. Then the reasoner will infer the following:

(Author → has_book → Book)

There are many plugins offered by Protégé to ease a task. One such plugin is Cellfie [12](Grocery Tutorial) plugin. This plugin helps in creation of OWL ontologies from excel or spreadsheet through a flexible mapping mechanism. We can create axioms and add individuals using this plugin.

b) Apache Jena Fuseki: Fuseki [13] is also an open-source semantic web framework based on Java which is used for running SPARQL query. We imported our ontology with inferenced axioms to run our complex and advanced queries.

c) OwlReady2: Owlready2 [14] is a Python library for doing ontology-oriented programming. It can load OWL 2.0 ontologies as Python objects, edit them, save them, and use HermiT to execute reasoning. Owlready2 enables OWL ontologies to be accessed in a transparent manner. We have used this to enhance our ontology and one can just use put their excel or spreadsheet and run our custom python script to load new data in AlloyOnto. The code for the same is mentioned in the Appendix section.

d) Pymatgen: It is an open-source python library for which can be used to extract various properties of materials by using APIs. We have used Pymatgen [15] library to get all the properties of elements like melting point, boiling point, thermal conductivity and many more.

CHAPTER 3: Methodology

We have built our AlloyOnto ontology from scratch. The main objective of AlloyOnto is to represent alloy related knowledge. To gain this knowledge we have exploit the elements and phase diagram properties. The general flow of creating the ontology is:

a) Identifying and defining the classes. We have defined Alloys, Elements, Phase Diagram Features and Phase Properties.
b) Defining data properties and object properties (between classes). Data properties links subject with some form of attribute data through predicate. While object properties links individuals of two classes through predicate.
c) Specify domain and range properties for every object properties.
d) Finally add individuals or instances to the relevant defined classes.
e) If a new classes need to be created then add data properties corresponding to that class and find relations of new class with existing classes and define them as object properties.

We have shown the detailed process through which we have created a valid and consistent OWL ontology:

3.1 Creating Elements class

First, we defined Elements class. Elements is the most fundamental class that we have defined. It consists of all the elements that are known to us. We have added both full name and their symbol as the individuals of Elements class. We captured the properties of elements using Pymatgen python library. The owl file is generated using the APIs of Pymatgen using python script file (code is shown in Appendix). In this owl file following prefixes are used:

```
@PREFIX owl: <http://www.w3.org/2002/07/owl#>
@PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

The namespace used for the individuals and properties in this ontology is as follows:

```
@PREFIX ae: <http://semantic.iitm.ac.in/AlloyOnto/Elements#>
```

The class defined to hold all the elements is with the IRI:

ae:Cu a ae:Elements

ae:Ni a ae:Elements

In addition, individuals with long names have also been created and mapped over to the corresponding individuals with symbol names using the relationship owl:sameAs.

ae:Copper owl:sameAs ae:Cu

ae:Nickel owl:sameAs ae:Ni

Table 2 – Data properties of Elements class with range and label

Data Property	Range	Label
ae:atomic_mass	xsd:float	Atomic mass
ae:atomic_orbitals	rdfs:Literal	Atomic orbitals as a string
ae:atomic_radius	xsd:float	Atomic radius in Angstroms
ae:block	rdfs:Literal	Block in the periodic table
ae:boiling_point	xsd:float	Boiling point in Kelvin
ae:brinell_hardness	xsd:float	Brinell hardness in units of MN/m^2
ae:bulk_modulus	xsd:float	Bulk modulus in units of GPa
ae:common_oxidation_states	rdfs:Literal	List of common oxidation states
ae:critical_temperature	xsd:float	Critical temperature
ae:electrical_resistivity	rdfs:Literal	Electrical resistivity with units as a string
ae:electronegativity	xsd:float	Pauling electronegativity
ae:group	rdfs:Literal	Group in the periodic table
ae:ionic_radii	rdfs:Literal	Ionic radii as a string

ae:is_actinoid	xsd:Boolean	Boolean value True if element is actinoid else False
ae:is_alkali	xsd:Boolean	Boolean value True if element is alkali else False
ae:is_alkaline	xsd:Boolean	Boolean value True if element is alkaline else False
ae:is_halogen	xsd:Boolean	Boolean value True if element is halogen else False
ae:is_lanthanoid	xsd:Boolean	Boolean value True if element is lanthanoid else False
ae:is_metalloid	xsd:Boolean	Boolean value True if element is metalloid else False
ae:is_noble_gas	xsd:Boolean	Boolean value True if element is noble_gas else False
ae:is_post_transition_metal	xsd:Boolean	Boolean value True if element is post_transition_metal else False
ae:is_rare_earth_metal	xsd:Boolean	Boolean value True if element is rare_earth_metal else False
ae:is_transition_metal	xsd:Boolean	Boolean value True if element is transition_metal else False
ae:max_oxidation_state	xsd:int	Maximum oxidation state
ae:melting_point	xsd:float	Melting point in Kelvin
ae:min_oxidation_state	xsd:int	Minimum oxidation state
ae:mineral_hardness	xsd:float	Mineral hardness
ae:number	xsd:integer	Atomic Number

ae:poisons_ratio	xsd:float	Poisson's ratio
ae:thermal_conductivity	xsd:float	Thermal conductivity in units of W/mK

3.2 Creating Alloys Class

We have defined an Alloys class which will consist of all types of alloys like binary alloys, ternary alloys etc. These binary alloys and ternary alloys will be the subclass of Alloys class. As of now we have only defined binary alloys due to data constraints but ternary alloys can also be added. We have added individuals of the Binary alloys class using the Cellfie plugin.

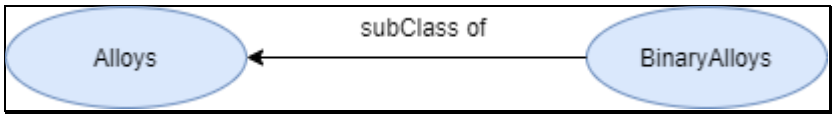

Fig 3 – Class BinaryAlloys is a subclass of Alloys4

We have defined object properties between Alloys and Elements class as follows:

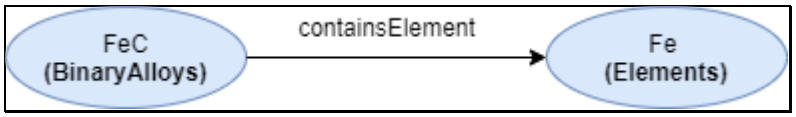

Fig 4 – FeC (Iron-Carbon) is an individual of class BinaryAlloys and Fe (Iron) is an individual of class Elements. So the above graph means that FeC binary alloy contains Fe element

And defined an inverse property elementContainedIn of containsElement which will give us:

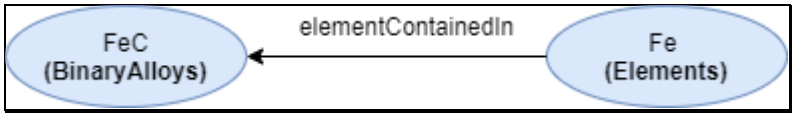

Fig 5 – FeC (Iron-Carbon) is an individual of class BinaryAlloys and Fe (Iron) is an individual of class Elements. So the above graph means that Fe element is contained in FeC.

3.3 Creating Phase Diagram Features Class

Next, we have defined the Phase_Diagram_Features class which has 3 subclasses to describe various properties of alloys extracted from phase diagrams. We have extracted data of phase diagrams for each binary alloy from the ASM database. Here we implemented the python script

owlReady2 library (code is shown in Appendix). The three subclasses are:

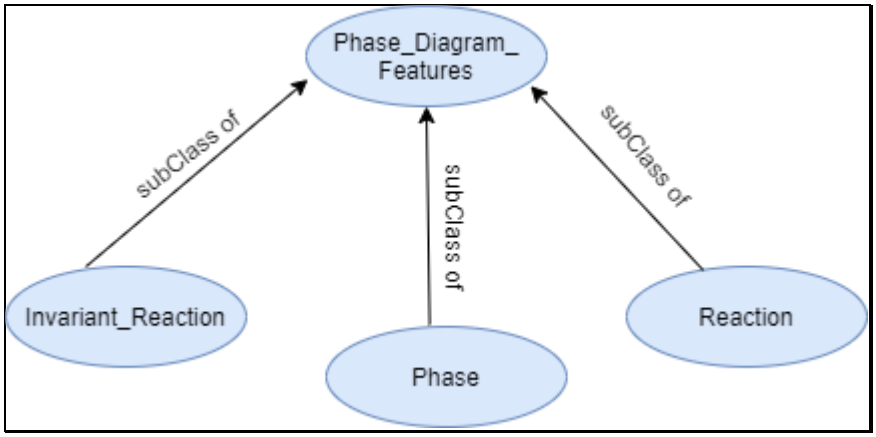

Fig 6 – Invariant_Reaction, Phase and Reaction are the three subclasses of class Phase_Diagram_Features.

Invariant_Reaction - This subclass contains types of invariant reactions that are possible in phases. We have taken Eutectic, Peritectic, Monotectic, Eutectoid and Peritectoid reactions as individuals. The object property that is associated with BinaryAlloy and Invariant_Reaction class is "hasReaction".

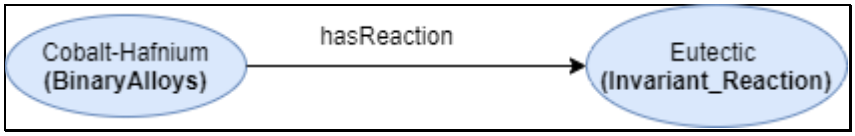

Fig 7 – Cobalt-Hafnium is an individual of BinaryAlloys and Eutectic is an individual of Invariant_Reaction. So the above graph means Cobalt-Hafnium has Eutectic reaction

We have defined the inverse property reactionIn of hasReaction which will give us:

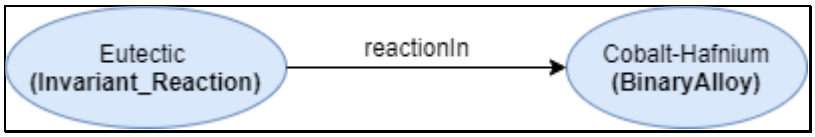

Fig 8 – Cobalt-Hafnium is an individual of BinaryAlloys and Eutectic is an individual of Invariant_Reaction. So the above graph means Eutectic reaction is present in Cobalt-Hafnium binary alloy

Phase - This subclass contains the phases that are present in alloys. The object property that is associated with BinaryAlloy and Phase class is "hasPhase".

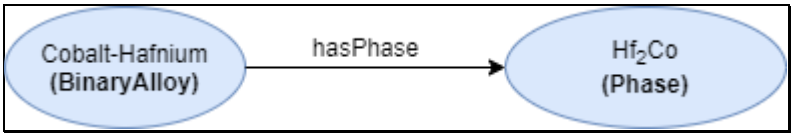

Fig 9 – Cobalt-Hafnium is an individual of BinaryAlloys and Hf_2Co is an individual of Phase. So the above graph means Cobalt-Hafnium binary alloy has Hf_2Co phase

We have defined the inverse property phaseIn of hasPhase which will give us:

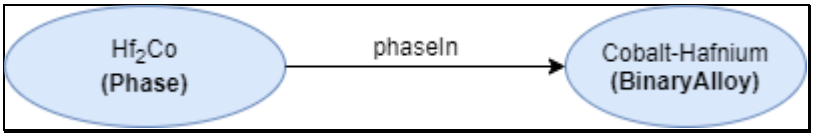

Fig 10 – Cobalt-Hafnium is an individual of BinaryAlloys and Hf_2Co is an individual of Phase. So the above graph means Hf_2Co phase is present in Cobalt-Hafnium binary alloys

Phase subclass also have some data properties.

Table 3 – Data properties of Phase class with range and label

Data Property	Range	Label
density_of_phase	xsd:decimal	Density of Phase as decimal
volume_of_phase	xsd:decimal	Volume of Phase as decimal

Reaction - This subclass contains the reaction of a binary alloy. There can be more than one reaction for a binary alloy that is why we have followed a syntax to represent them that is "<Alloy Symbol>_R<Reaction number>. One example above is AgHf_R3 which represents reaction number 3 for alloy Gold-Hafnium. We have defined the rdfs:label to link exact reaction with syntax we have used earlier. This subclass is associated with 2 object properties that are is_type and containsReaction.

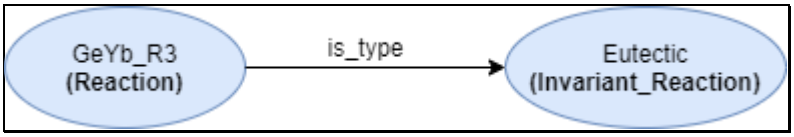

Fig 11 – GeYb_R3 is an individual of Reaction and Eutectic is an individual of Invariant_Reaction. So the above graph means GeYb_R3 reaction is a type of Eutectic reaction where GeYb_R3 represents "L2 + $Yb_3Ge_{4.3}$ rt = $Yb_{10}Ge_{11}$ reaction

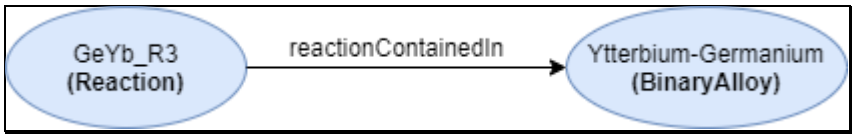

Fig 12 – GeYb_R3 is an individual of Reaction and Ytterbium-Germanium is an individual of BinaryAlloy. So the above graph means GeYb_R3 reaction is a contained in Ytterbium-Germanium

We have defined inverse property containsReaction of reactionContainedIn which will give us:

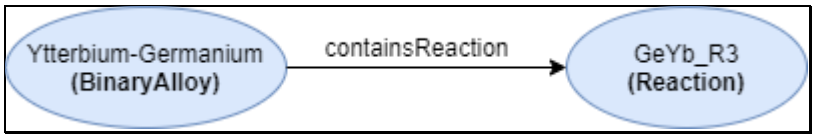

Fig 13 – GeYb_R3 is an individual of Reaction and Ytterbium-Germanium is an individual of BinaryAlloy. So the above graph means Ytterbium-Germanium contains GeYb_R3 reaction

3.4 Creating Phase Properties Class

This class tells us about the properties of a phase present in an alloy. This class contains 2 subclasses which are:

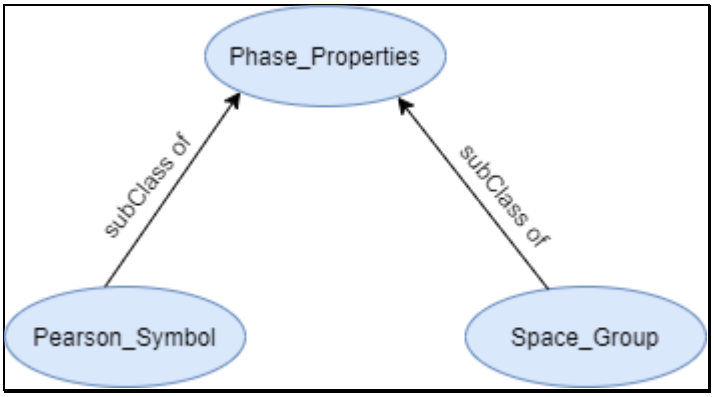

Fig 14 – Pearson_Symbol and Space_Group are the two subclasses of class Phase_Properties

Pearson_Symbol - Pearson symbol is used to describe crystal structure. This subclass contains all the possible pearson symbols. This subclass is associated with one object property which is hasPearsonSymbol. It gives us the phases which have this pearson symbol.

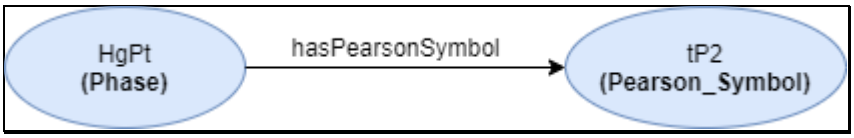

Fig 15 – HgPt is an individual of Phase and tP2 is an individual of Pearson_Symbol. So the above graph means HgPt Phase has tP2 pearson symbol

We have defined inverse property pearsonSymbolIn of hasPearsonSymbol which will give us:

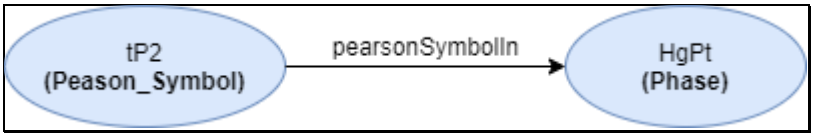

Fig 16 – HgPt is an individual of Phase and tP2 is an individual of Pearson_Symbol. So the above graph means tP2 pearson symbol present in HgPt Phase

Space_Group - Space group is symmetry group of a configuration in 3D space. This subclass contains all the possible space groups. This subclass is associated with one object property which is hasSpaceGroup. It gives us the phases which have this space group.

Fig 17 – HgPt is an individual of Phase and P4/mmm is an individual of Space_Group. So the above graph means HgPt Phase has P4/mmm space group

We have defined inverse property spaceGroupIn of hasSpaceGroup which will give us:

Fig 18 – HgPt is an individual of Phase and P4/mmm is an individual of Space_Group. So the above graph means P4/mmm space group is present in HgPt Phase

3.5 Hierarchy and Querying data

We have added all the classes and their individuals, object properties and data properties and we get a complete hierarchy. This hierarchy gives are relations between different classes and much more.

Fig 19 – This graph is the complete hierarchy of our ontology which shows the relation between the classes. At the bottom right corner, meaning of each shape is mentioned

Table 4 – All Object properties in AllyOnto with domain, range and inverse property

Object Propery	**Domain**	**Range**	**Inverse Property** (if any)
containsElement	Alloys	Element	elementContainedIn
hasPhase	BinaryAlloy	Phase	phaseContainedIn
containsReaction	BinaryAlloy	Reaction	reactionContainedIn
hasPearsonSymbol	Phase	Pearson_Symbol	pearsonSymbolIn
hasReaction	BinaryAlloy	Invariant_Reaction	reactionIn
hasSpaceGroup	Phase	Space_Group	spaceGroupIn
is_type	Reaction	Invariant_Reaction	-

After designing and building this ontology we need to query the data such that we can get the desired results. We have used SPARQL query language for this purpose. SPARQL is an RDF query language- that is, a semantic query language for databases and also able to retrieve and manipulate data stored in RDF format.

CHAPTER 4: Results and Discussion

Based on our ontology and knowledge graph, we have written SPARQL queries for the questions asked in formal language and the results we get from the reasoner. The prefixes that we have used are shown below:-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ae: <http://semantic.iitm.ac.in/AlloyOnto/Elements#>
PREFIX a: <http://semantic.iitm.ac.in/AlloyOnto#>
```

1) **List binary alloys and corresponding phases with Pearson symbol cF4 and contains Copper as an element.**

Query -

```
SELECT ?x ?y
        WHERE {
                ?x a:hasPearsonSymbol a:cF4 .
                ?y a:hasPhase ?x .
                ?y a:containsElement ae:Cu
        }
```

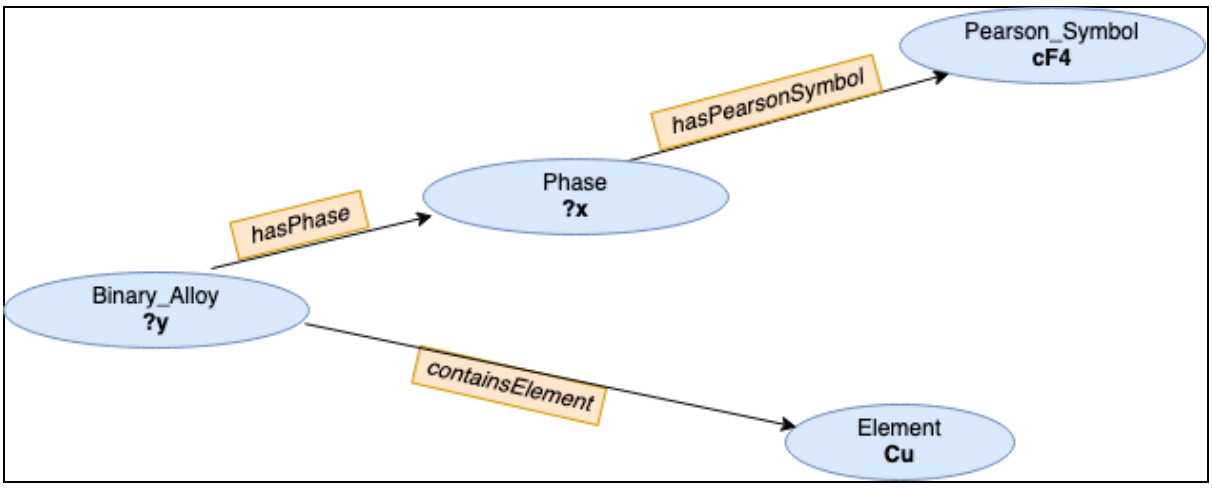

Fig 20 – Graph representation of SPARQL query 1

Output –

Table 5 – Query 1

?x (Phase)	?y (BinaryAlloy)
Cu0.5Au0.5	CuAu
Cu0.95Ti0.05	CuTi

Answer - CuAu is a alloy has phase Cu0.5Au0.5 which contains copper and pearson symbol of cF4

2) **List binary alloy and it's corresponding phases that have phase density more than 20 gm/cc at room temperature**

Query -

```
SELECT ?x ?y ?a
WHERE {
        ?x a:density_of_phase ?y .
        ?x rdfs:label ?z .
        FILTER(?y >= 20) .
        FILTER regex(?z, "rt", "i") .
        ?a a:hasPhase ?x
}
```

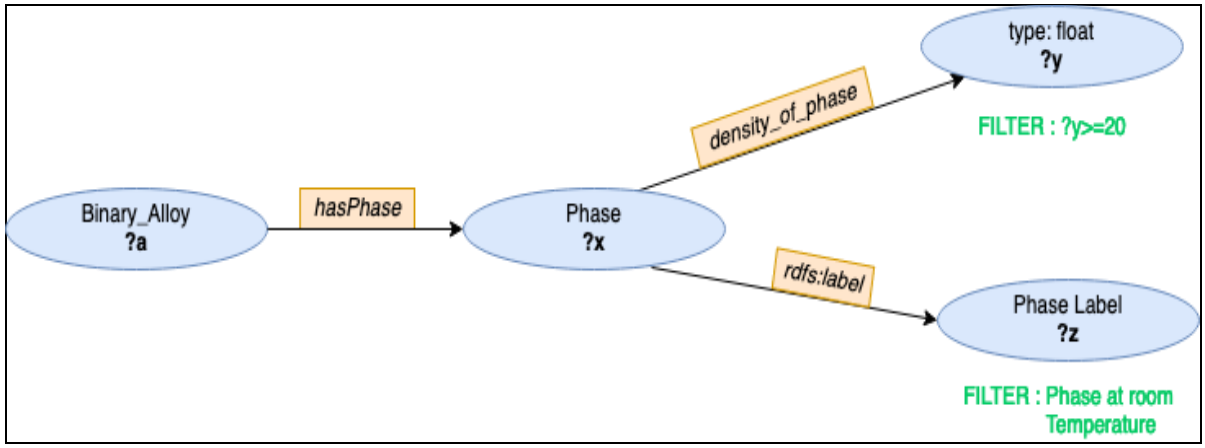

Fig 21 – Graph representation of SPARQL query 2

Output –

Table 6 – Query 2

?x (Phase)	?y (density g/cc)	?a (BinaryAlloy)
W0.25Re0.75	20.22	WRe

Answer - WRe alloy has a phase W0.25Re0.75 which has density 20.22 g/cc at room temperature.

3) List the alloys which contain gamma-Fe phases which has monotectic invariant reaction.

Query -

```
SELECT ?b
WHERE {
        ?x rdfs:comment ?z .
        FILTER regex(?z, "gamma;Fe", "i") .
        ?x a:phaseContainedIn ?b.
        ?b a:hasReaction a:Monotectic
}
```

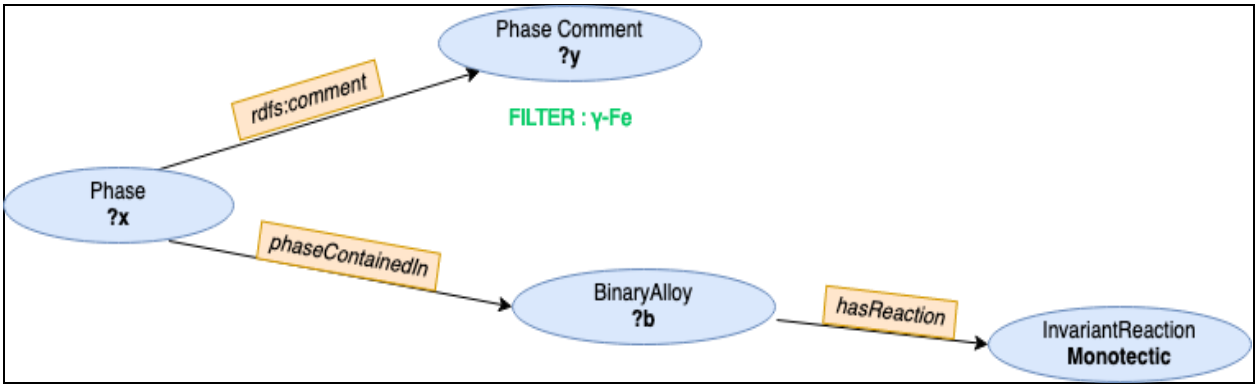

Fig 22 – Graph representation of SPARQL query 3

Output –

Table 7 – Query 3

?b (Alloy)
FeSn

Answer - FeSn is an alloy which contains γ-Fe and has Monotectic invariant reaction.

4) List alloys that have eutectic reaction above 1400 K and contains Nickel

Query -

```
SELECT DISTINCT ?x ?a ?z
        WHERE {
                ?x a:containsElement ae:Ni .
                ?x a:containsReaction ?y .
                ?y a:is_type a:Eutectic .
                ?y a:at_temperature ?z.
                ?y rdfs:label ?a .
                FILTER(?z>=1400) .
        }
```

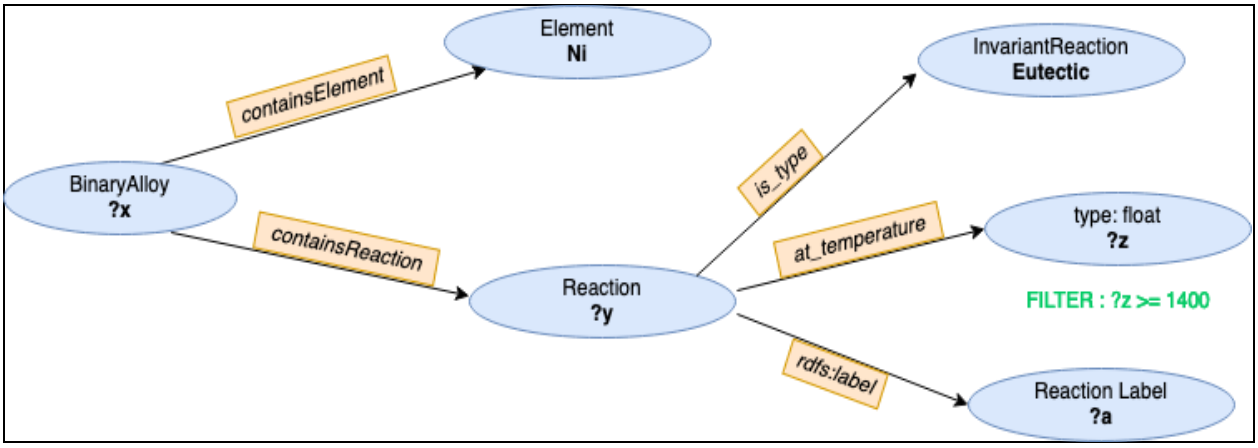

Fig 23 – Graph representation of SPARQL query 4

Output –

Table 8 – Query 4

?x (Alloy)	?a (Reaction)	?z (Temperature)
NiW	L = (Ni) + (W)	1495

Answer - NiW is an alloy which contains eutectic reaction { L = (Ni) + (W) } at temperature 1495 K and contains Nickel element

5) **List alloys that have peritectic, peritectoid, and monotectic.**

Query-

```
SELECT ?x
WHERE {
        ?x a:hasReaction a:Peritectic .
        ?x a:hasReaction a:Peritectoid .
        ?x a:hasReaction a:Monotectic .
}
```

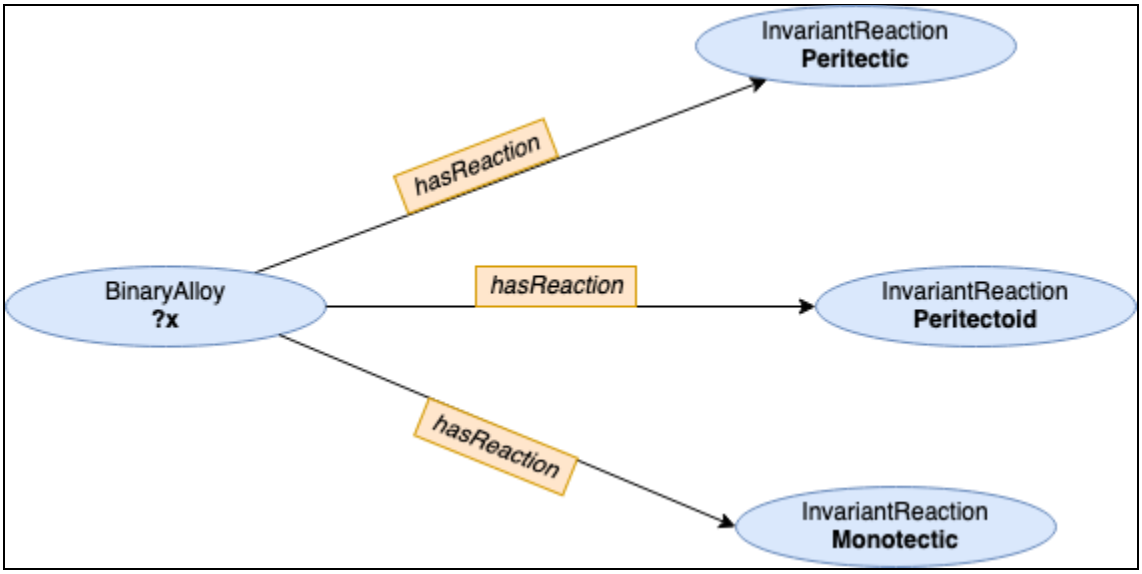

Fig 24 – Graph representation of SPARQL query 5

Output –

Table 9– Query 5

?x (Alloy)
TeCr

Answer - TeCr is an alloy which contains peritectic, peritectoid, and monotectic reaction.

CHAPTER 5: Summary and Conclusions

Alloy selection and their application necessitate a high level of material understanding. In light of the huge amount of alloys available, it is critical to investigate and design an open, formal, and shared knowledge representation and framework for content selection.

In classification, sharing, and formalization, ontology-based Semantic Web technologies have several advantages over conventional idea modeling methods, and they offer a promising answer to present difficulties.

In this thesis, we started with defining fundamental concepts and classes and derived relations between different classes. We have described the data properties that an individual of class can possess. There are many ways to generate or modify an ontology (OWL file) and here we have shown two methods that can be used for this. First one can use Protégé software which is open source and user-friendly. It provides various plugins, which makes a developer work easy and it does not require any sort of coding knowledge. Secondly, we have used owlready2 library which is a python package to generate or modify ontology. This is a very efficient way because one can easily modify ontology using this by changing a few lines of code.

To evaluate our ontology we have used SPARQL queries to answer complex and advanced questions which can't be answered by web search engines or databases. To write a SPARQL query, one should understand the hierarchy of the ontology which will help to know the relations that exist between different concepts.

5.1 Prospects Generated by this Work:

1. We have captured the semantics of the data which help us to represent and understand the materials domain. Visualization of data is also very easy here by generating knowledge graphs.
2. The alloy data is distributed in many resources but none of them gives us complete knowledge about them. Through our ontology, we have tried to integrate this data from

heterogeneous resources which can help a user to get every small detail related to alloys.

3. The most important issue that this paper addresses is to retrieve relevant results to a user for a complex and advanced question which requires understanding of more than one concept.

5.2 Recommendations for Future Work:

1. The most important feature of ontology is reasoning ability and to do that one should have as many rules and axioms as possible. Although we have defined sufficient rules, one can add more rules in it to improve this ontology.
2. As we have discussed earlier, a lot of data is generated from heterogeneous resources which may not be digital or accessible. So this makes it difficult to collect as this will require a lot of manual work and domain knowledge. Here NLP techniques can be used to extract relevant information from research papers and books but this will require a very robust algorithm.
3. Instead of writing a query in SPARQL, we can devise an algorithm that can translate the English sentences into SPARQL query using NLP techniques [16].

Appendix

Code File 1 - To extract properties of binary alloys like elements, reaction, reaction type and phases from excel sheets of different binary alloys and combine them into single excel file.

```
# -*- coding: utf-8 -*-
"""Extract Phases

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1hgz7XnKs12aC1Op_4m07IIDIu5o54Bxf
"""

import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
import re
from os import listdir, makedirs, removedirs
from google.colab import drive
from os.path import join, exists, isdir, isfile
drive.mount('/content/gdrive')

data_dir = "/content/gdrive/My Drive/BTP- Materials Ontology/Phase_Diagram_2"

binary_names, file_paths = [], []
for binary_name in sorted(listdir(data_dir)):
  folder_path = join(data_dir,binary_name)
  if not isdir(folder_path):
    continue
  paths = [join(folder_path, f) for f in listdir(folder_path) if isfile(join(fol
der_path, f))]
  # n_pictures = len(paths)
  # person_name = person_name.replace('_', ' ')
  binary_names.extend([binary_name])
  file_paths.extend(paths)

new_binary_name = list()
for i in range(len(binary_names)):
  temp = binary_names[i].split("Binary")
  binary_names[i] = temp[0]

print(binary_names[:5], file_paths[:5])

# To extract reactions and phases present in the reaction of binary systems
def make_data(binary_names, file_paths):
  n = len(file_paths)
  binary_systems = list()
```

```
invariant_reactions = list()
element1 = list()
element2 = list()
temperature = list()
complete_reaction = list()
complete_reaction_2 = list()
comp_a = list()
comp_b = list()
comp_c = list()
phase_1 = list()
phase_2 = list()
phase_3 = list()

for i in range(n):

  # Accessing "Reaction-Data" sheet in excel file of each binary system
  xls = pd.ExcelFile(file_paths[i])
  df = pd.read_excel(xls, 2)

  l1 = []
  l2 = []
  l3 = []
  l4 = []
  l5 = []
  l6 = []
  l7 = []
  l8 = []
  l9 = []
  l10 = []
  l11 = []
  l12 = []
  l13 = []

  l2 = list(df.iloc[6:,12])
  # l2 = list(set(l2))
  l5 = list(df.iloc[6:,11])
  # Repeat it for composition and Complete reaction
  l6 = list(df.iloc[6:,3])
  l7 = list(df.iloc[6:,4])
  l8 = list(df.iloc[6:,6])
  l9 = list(df.iloc[6:,8])
  # Remove unknown keyword if present in invariant reaction
  # if l2[-1] == "unknown":
  #   l2 = l2[:len(l2)-1]

  l1 = list()
  for j in range(len(l2)):
    l1.append(binary_names[i])
    x,y = binary_names[i].split("-")
    l3.append(x)
```

```
      l4.append(y)

    binary_systems += l1
    invariant_reactions += l2
    element1 += l3
    element2 += l4
    temperature += l5
    complete_reaction += l6
    comp_a += l7
    comp_b += l8
    comp_c += l9

    for k in range(len(complete_reaction)):
      p,q = complete_reaction[k].split("\n")
      # q = q.replace(" ","")
      l10.append(q)
      s = re.split(" = | \\+ ",q)
      if len(s) == 3:
        a,b,c = s
      elif len(s) == 2:
        a,b = s
        c = "NULL"
      else:
        a = s
        b,c = "NULL", "NULL"
      l11.append(a)
      l12.append(b)
      l13.append(c)
  complete_reaction_2 = l10
  phase_1 = l11
  phase_2 = l12
  phase_3 = l13



  # Create New Dataframe to store all binary systems and their invariant reactio
ns
  new_df = pd.DataFrame(list(zip(binary_systems,invariant_reactions,element1,ele
ment2,temperature, complete_reaction_2, comp_a, comp_b, comp_c, phase_1, phase_2
, phase_3)), columns=["Binary System","Invariant Reaction","Element1","Element2"
, "Temperature", "Reaction", "Composition A", "Composition B", "Composition C",
"Phase 1", "Phase 2", "Phase 3"])
  return new_df

final_df = make_data(binary_names, file_paths)

reactions = ['Monotectic','Eutectic','Eutectoid','Peritectic','Peritectoid']
indexes = list()
for i in range(final_df.shape[0]):
  if final_df.iloc[i,1] not in reactions:
```

```
    indexes.append(i)

final_df.drop(indexes, inplace=True)

final_df[:10]

len(final_df)

!pip install periodictable
!pip install pymatgen==2021.3.3

from periodictable import elements
from pymatgen import Element as El

Elements = list()
gen = (el for el in elements if (el.number < 108 and el.number > 0))
for el in gen:
    myel = El(el.symbol)
    Elements.append([myel.name, myel.long_name])
len(Elements)

from collections import defaultdict
Elements_dict = defaultdict()
for i in range(len(Elements)):
  Elements_dict[Elements[i][1]] = Elements[i][0]

reaction_name1 = list()
reaction_name2 = list()

j = 1
for i in range(0,len(final_df)):
  if i == 0:
    reaction_name1.append(Elements_dict[final_df.iloc[i][2]] + Elements_dict[fin
al_df.iloc[i][3]] + "_R" + str(j))
    reaction_name2.append(Elements_dict[final_df.iloc[i][3]] + Elements_dict[fin
al_df.iloc[i][2]] + "_R" + str(j))

  else:
    if final_df.iloc[i][0] == final_df.iloc[i-1][0]:
      j = j+1
      reaction_name1.append(Elements_dict[final_df.iloc[i][2]] + Elements_dict[f
inal_df.iloc[i][3]] + "_R" + str(j))
      reaction_name2.append(Elements_dict[final_df.iloc[i][3]] + Elements_dict[f
inal_df.iloc[i][2]] + "_R" + str(j))

    else:
      j = 1
      reaction_name1.append(Elements_dict[final_df.iloc[i][2]] + Elements_dict[f
inal_df.iloc[i][3]] + "_R" + str(j))
```

```
      reaction_name2.append(Elements_dict[final_df.iloc[i][3]] + Elements_dict[f
inal_df.iloc[i][2]] + "_R" + str(j))

len(reaction_name1)

final_df['Reaction Name 1'] = reaction_name1
final_df['Reaction Name 2'] = reaction_name2
final_df[:5]

path = "gdrive/My Drive/BTP- Materials Ontology/Phases_reaction.xlsx"

final_df.to_excel(path)
```

Code File 2 – To extract properties of Phase like Pearson symbol and space group of phases present in binary alloys from excel sheets of different binary alloys and combine them into single excel file.

```
# -*- coding: utf-8 -*-
"""Phases Properties.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/102JOKL8ONJXLhNEMNTYS9yW7uInY8Wfs
"""

import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
import re
from os import listdir, makedirs, removedirs
from google.colab import drive
from os.path import join, exists, isdir, isfile
drive.mount('/content/gdrive')

data_dir = "/content/gdrive/My Drive/BTP- Materials Ontology/Phase_Diagram_2"

binary_names, file_paths = [], []
for binary_name in sorted(listdir(data_dir)):
  folder_path = join(data_dir,binary_name)
  if not isdir(folder_path):
    continue
  paths = [join(folder_path, f) for f in listdir(folder_path) if isfile(join(fol
der_path, f))]
  # n_pictures = len(paths)
  # person_name = person_name.replace('_', ' ')
  binary_names.extend([binary_name])
```

```
  file_paths.extend(paths)

new_binary_name = list()
for i in range(len(binary_names)):
  temp = binary_names[i].split("Binary")
  binary_names[i] = temp[0]

n = len(file_paths)
binary_systems = list()
compound = list()
density = list()
pearson_symbol = list()
volume = list()
phase_label = list()
for i in range(n):
  xls = pd.ExcelFile(file_paths[i])
  df = pd.read_excel(xls, 0)
  l1 = []
  l2 = []
  l3 = []
  l4 = []
  l5 = []
  l6 = []

  l2 = list(df.iloc[6:,4])
  l3 = list(df.iloc[6:,7])
  l4 = list(df.iloc[6:,6])
  l5 = list(df.iloc[6:,8])
  l6 = list(df.iloc[6:,3])

  for j in range(len(l2)):
    l1.append(binary_names[i])

  binary_systems += l1
  compound += l2
  density += l3
  pearson_symbol += l4
  volume += l5
  phase_label += l6


  # for k in range(len(compound)):
new_df = pd.DataFrame(list(zip(binary_systems,compound,phase_label,density,pears
on_symbol,volume)), columns = ["Binary", "Compound", "Phase label", "Density", "
Pearson Symbol", "Volume"])

m = len(new_df)
Y = list()
X = list()
Z = list()
```

```
U = list()
V = list()
for i in range(m):
  y = str(new_df.iloc[i,1]).split("\n")
  y[-1] = y[-1].replace("~","")
  Y.append(y[-1])
  L = str(new_df.iloc[i,4]).split("\n")
  if len(L) == 3:
    x,z = L[1],L[2]
    x = x.replace("\r","")
    z = z.replace("~","")
  else:
    x,z = float("NaN"), float("NaN")
  X.append(x)
  Z.append(z)
  M = str(new_df.iloc[i,2]).split("\n")
  # print(len(M))
  if len(M)==4:
    u,v = M[1],M[3]
    u = u.replace("~","")
    v = v.replace("~","")
  U.append(u)
  V.append(v)

new_df['Formula'] = Y
new_df['Pearson_Symbol'] = X
new_df['Space Group'] = Z
new_df['APD Phase label'] = U
new_df['Published Phase label'] = V

new_df.drop(['Compound','Pearson Symbol', 'Phase label'], axis=1, inplace = True
)

new_df.dropna(axis = 0, inplace = True)

new_df[:5]

path = "gdrive/My Drive/BTP- Materials Ontology/Phases_Properties_2.xlsx"
new_df.to_excel(path)
```

Code File 3 – Modify owl file using owlready2 python library. This code try to cover exhaustive list of operations that are commonly required to make or modify owl file

```
from owlready2 import *
import numpy as np
import pandas as pd
import os
```

```
## Specify the path for existing owl file
onto = get_ontology("file:///Users/tapishgarg/Documents/BTP-
Ontologies/Extending_AlloyOnto/AlloyOnto1.owl")
onto.load()

print(onto.get_namespace(onto.base_iri))
print(list(onto.classes()))
print(onto.imported_ontologies)

## Import excel sheets
data = pd.read_excel("/Users/tapishgarg/Documents/BTP-
Ontologies/Extending_AlloyOnto/Phases_Properties_2.xlsx")
data_2 = pd.read_excel("/Users/tapishgarg/Documents/BTP-
Ontologies/Extending_AlloyOnto/Phases_reaction.xlsx")

enamespace = onto.get_namespace(onto.base_iri)

## Create Class

class Phase_Properties(Thing):
    namespace = onto


## Create Subclass

class Pearson_Symbol(Phase_Properties):
    pass

class Space_Group(Phase_Properties):
    pass


## Create Object Properties

with onto:
    class hasPearsonSymbol(ObjectProperty):
        domain = [onto.Phase]
        range = [onto.Pearson_Symbol]
        pass

with onto:
    class hasSpaceGroup(ObjectProperty):
        domain = [onto.Phase]
        range = [onto.Space_Group]


## Create Inverse Properties

with onto:
    class pearsonSymbolIn(ObjectProperty):
```

```
        domain = [onto.Pearson_Symbol]
        range = [onto.Phase]
        inverse_property = hasPearsonSymbol

with onto:
    class spaceGroupIn(ObjectProperty):
        domain = [onto.Space_Group]
        range = [onto.Phase]
        inverse_property = hasSpaceGroup


## Create Data Properties

with onto:
    class density_of_phase(DataProperty):
        domain = [onto.Phase]
        range = [float]

    class volume_of_phase(DataProperty):
        domain = [onto.Phase]
        range = [float]


## Create Individuals for a class

with onto:
    for i in range(data.shape[0]):
        pearson_symbol = onto.Pearson_Symbol(data["Pearson_Symbol"][i])
    # pearson_symbol = onto.Pearson_Symbol()

    for i in range(data.shape[0]):
        space_group = onto.Space_Group(data["Space Group"][i])


## Creating Axioms

with onto:
    for i in range(data.shape[0]):
        individual = onto.Phase(data['Formula'][i])
        individual.hasPearsonSymbol.append(onto.Pearson_Symbol(data['Pearson_Sym
bol'][i]))
        individual.hasSpaceGroup.append(onto.Space_Group(data['Space Group'][i])
)
        individual.density_of_phase.append(float(data['Density'][i]))
        individual.volume_of_phase.append(float(data['Volume'][i]))

onto.save(file="./AlloyOnto2.owl", format = "rdfxml")
```

References

[1] Y. Zhang, X. Luo, Y. Zhao and H.-c. Zhang, "An ontology-based knowledge framework for engineering material selection," *Advanced Engineering Informatics,* vol. 29, p. 985–1000, 2015.

[2] "Online Databases - ASM International," [Online]. Available: https://www.asminternational.org/materials-resources/online-databases.

[3] E. A. Brandes and G. B. Brook, Smithells metals reference book, Elsevier, 2013.

[4] "SPARQL Query Language for RDF," [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/.

[5] f. given=AH, f. given=J, f. given=M, f. given=T, f. given=G and f. given=RJ, "A semantic modeling approach for the automated detection and interpretation of structural damage," *Automation in Construction,* vol. 128, p. 103739, 2021.

[6] X. Zheng, B. Wang, Y. Zhao, S. Mao and Y. Tang, "A knowledge graph method for hazardous chemical management: Ontology design and entity identification," *Neurocomputing,* vol. 430, p. 104–111, 2021.

[7] Y. Duan, L. Hou and S. Leng, "A novel cutting tool selection approach based on a metal cutting process knowledge graph," *The International Journal of Advanced Manufacturing Technology,* vol. 112, p. 3201–3214, 2021.

[8] P. E. van der Vet, P.-H. Speel and N. J. I. Mars, "The Plinius ontology of ceramic materials," in *Eleventh European Conference on Artificial Intelligence (ECAI'94) Workshop on Comparison of Implemented Ontologies*, 1994.

[9] A. Radinger, B. Rodriguez-Castro, A. Stolz and M. Hepp, "BauDataWeb: the Austrian building and construction materials market as linked data," in *Proceedings of the 9th International Conference on Semantic Systems*, 2013.

[10] M. A. Musen, "The protégé project: a look back and a look forward," *AI Matters,* vol. 1, p. 4–12, 2015.

[11] f. given=S.C.F.B.I., "protégé," [Online]. Available: https://protege.stanford.edu/.

[12] "Protege - Cellfie plugin," [Online]. Available: https://github.com/protegeproject/cellfie-plugin/wiki/Grocery-Tutorial.

[13] "Apache Jena - Apache Jena Fuseki," [Online]. Available: https://jena.apache.org/documentation/fuseki2/.

[14] J.-B. Lamy, "Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies," *Artificial intelligence in medicine,* vol. 80, p. 11–28, 2017.

[15] "Introduction — pymatgen 2022.0.8 documentation," [Online]. Available: https://pymatgen.org/.

[16] P. Ochieng, "PAROT: Translating natural language to SPARQL," *Expert Systems with Applications: X,* vol. 5, p. 100024, 2020.