Module 4 - Critical Thinking Assignment: Taxonomy

Alejandro Ricciardi

Colorado State University Global

CSC410: Artificial Intelligence

Dr. Chris Whitfield

November 3, 2024

Module 4 - Critical Thinking Assignment: Taxonomy

Taxonomy in computer science is the act of classifying and organizing concepts. For example, in software engineering, it is used to classify software testing techniques, model-based testing approaches, and static code analysis tools (Novak et al., 2010). In data management, it is used to organize metadata and categorize-manage data assets (Knight, 2021). In Artificial Intelligence it is used to guide models to recognize patterns in data sets.

Taxonomy can also be defined as the act of organizing knowledge within a domain by using a controlled vocabulary to make it easier to find related information (Knight, 2021), and it must:

- “Follow a hierarchic format and provide names for each object in relation to other objects.
- May also capture the membership properties of each object in relation to other objects.
- Have specific rules used to classify or categorize any object in a domain. These rules must be complete, consistent, and unambiguous.
- Apply rigor in specification, ensuring any newly discovered object must fit into one and only one category or object.
- Inherit all the properties of the class above it but can also have additional properties.”

(Knight, 2021, p.1).

In this paper, taxonomic knowledge and frames are implemented in the domain of Programming Languages, focusing on Python. “A frame is a data structure that can represent the knowledge in a semantic net” (Colorado State University Global, n.d., p.2). To implement the taxonomic knowledge the paper follows three steps using first-order logic. The steps are Subset Information, Set Membership of Entities, and Properties of Sets and Entities. Then, the paper uses a tree-like structure to show how subcategories relate to parent categories. Additionally, it demonstrates how the hierarchical taxonomic structure interacts with frames, by illustrating how attributes and properties are defined in the Python frame and they align with the broader taxonomic categories. Finally, it explains how the combination of taxonomic relationships and frames provides a comprehensive representation of knowledge.

The Three Steps to Implement Taxonomic Knowledge

Note that the programming languages Java and Python are used as examples.

Step 1: Subset Information

In this step to represent the subcategory, first-order logic is used.

1. **Subcategory Relationships:**
   - **Compiled Languages and Interpreted Languages:**

$$\forall x\, High\_Level\_Compiled\_Language(x) \Rightarrow Compiled\_Language(x)$$

$$\forall x\, Scripting\_Language(x) \Rightarrow Interpreted\_Language(x)$$

   - **Programming Languages:**

$$\forall x\, Compiled\_Language(x) \Rightarrow Programming\_Language(x)$$

$$\forall x\, Interpreted\_Language(x) \Rightarrow Programming\_Language(x)$$

   - **specific Languages:**

$$\forall x\, Java(x) \Rightarrow High\_Level\_Compiled\_Language(x)$$

$$\forall x\, Python(x) \Rightarrow Scripting\_Language(x)$$

2. **Additional Subcategories:**
   - **Functional Languages and Logic Programming Languages:**

$$\forall x\, Functional\_Language(x) \Rightarrow Programming\_Language(x)$$

$$\forall x\, Logic\_Programming\_Language(x) \Rightarrow Programming\_Language(x)$$

Step 2: Set Membership of Entities

In this step, the category membership of specific languages is represented.

1. **Instances of Programming Languages:**
   - **Java SE 23:**

$$Java(JavaSE23)$$

   - **Python 3.13:**

$$Python(Python3\_13)$$

2. **Other Programming Languages (more examples):**

$$C(C23)$$

$$Haskell(Haskell2010)$$

Step 3: Properties of Sets and Entities

In this step, the properties of the categories and language program are represented.

1. **Properties of Programming Languages:**

- All Programming Languages have Syntax and are used for Software Development:

$$\forall x\, Programming\_Language(x) \Rightarrow Has\_Syntax(x)$$

$$\forall x\, Programming\_Language(x) \Rightarrow Used\_For(x, "Software\_Development")$$

2. **Properties of Compiled and Interpreted Languages:**
    - Compiled Languages have Execution Model 'Compiled'**:**

$$\forall x\, Compiled\_Language(x) \Rightarrow Execution\_Model(x, "Compiled")$$

    - Interpreted Languages have Execution Model 'Interpreted':

$$\forall x\, Interpreted\_Language(x) \Rightarrow Execution\_Model(x, "Interpreted")$$

3. **Properties of Specific Languages:**
    - Java has Static Typing Discipline:

$$\forall x\, Java(x) \Rightarrow Typing\_Discipline(x, "Static")$$

    - Python has Dynamic Typing Discipline:

$$\forall x\, Python(x) \Rightarrow Typing\_Discipline(x, "Dynamic")$$

    - Java Supports Paradigm 'Object-Oriented':

$$\forall x\, Java(x) \Rightarrow Supports\_Paradigm(x, "Object - Oriented")$$

    - Python Supports Multiple Paradigms:

$$\forall x\, Python(x) \Rightarrow Supports\_Paradigm(x, "Multi - Paradigm")$$

4. **Properties of Entities:**
    - **Java SE 23's Latest Version is 23:**

$$Latest\_Version(JavaSE23, "23")$$

    - **Python 3.13's Latest Version is 3.13:**

$$Latest\_Version(Python3\_13, "3.13")$$

Hierarchical Taxonomy of Programming Languages

Below is a (text shorter version) tree-like hierarchical structure representing the relationships between different languages in the domain of programming languages:

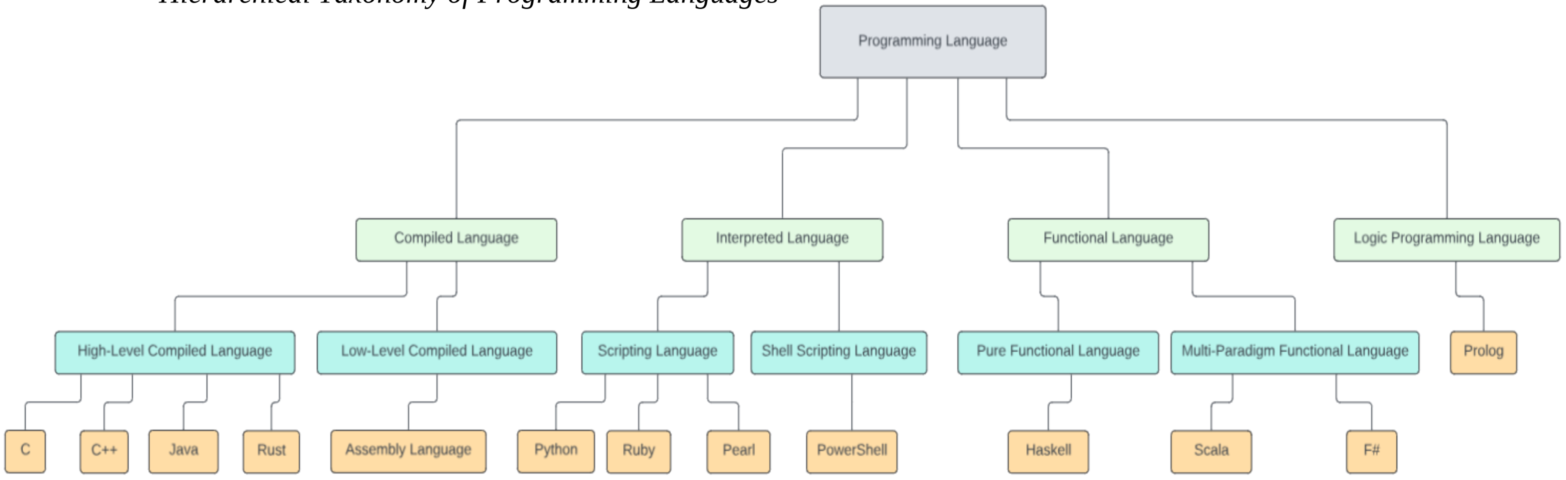
- Programming Language
    - Compiled Language
        - High-Level Compiled Language
            - C
            - C++
            - Java
            - Rust
        - Low-Level Compiled Language
            - Assembly Language

- Interpreted Language
  - Scripting Language
    - Python
    - Ruby
    - Perl
  - Shell Scripting Language
    - Bash
    - PowerShell
- Functional Language
  - Pure Functional Language
    - Haskell
  - Multi-Paradigm Functional Language
    - Scala
    - F#
- Logic Programming Language
  - Prolog

Note that some languages, like Python and Java, can be considered both interpreted and compiled languages; however, for the scope of this exercise, they are categorized as interpreted and compiled languages, respectively.

Hierarchical Taxonomy Visualization

Figure 1
Hierarchical Taxonomy of Programming Languages



Note: The diagram is a visual representation of the hierarchical taxonomy of programming languages. Data adapted from multiple sources: (Epözdemir, 2024; Foster, 2013; Gómez, n.d.; Peter Van Roy, 2008; Saxena, 2024; Startups, 2018; & Wikipedia contributors, 2024)

Specific Frame Python Interaction With Hierarchical Taxonomy

This section illustrates a Python frame, which is a data structure representation of the Python programming language attributes and properties. For comparison, a Java frame is also provided.

Figure 2

Python Frame

```
)
       Python
       Instance_Of: Scripting_Language;

       // Inherited properties and attributes
       Used_For: Software_Development
       Execution_Model: Interpreted
       Syntax: Easy_To_Use;

       // Properties and attributes specific to Python
       Creator: Guido van Rossum;
       First_Released: 1991;
       Typing_Discipline: Dynamic, Strong Typing;
       Paradigms: Object-Oriented, Imperative, Functional, Procedural, Reflective;
       License: Python Software Foundation License;
       Latest_Version: 3.13;
       Official_Website: www.python.org
)
```

Note: This is a frame representation of Python's properties and attributes. An example of an attribute is '`Instance_Of`' and of a property is '`Scripting_Language`'.

For comparison, below is a representation of the Java frame.

Figure 3

Java Frame

```
)
       Java
       Instance_Of: High_Level_Compiled_Language;

       // Inherited properties and attributes
       Used_For: Software_Development
       Debugging: Friendly
       Execution_Model: Compiled

       // Properties and attributes specific to Java
       Creator: James Gosling;
       First_Released: 1995;
       Typing_Discipline: Static, Strong Typing;
       Paradigms: Object-Oriented, Class-based, Concurrent;
       License: GNU General Public License with Classpath Exception;
       Latest_Version: 23;
       Official_Website: www.oracle.com/java/
)
```

Note: This is a frame representation of Java's properties and attributes.

A hierarchical taxonomy organizes entities into a tree-like structure. In the Programming Language hierarchical taxonomy, the root class (category representing the domain) or the first node of the tree-like structure is '*Programming Language,*' with all other nodes as subclasses (subcategories) that inherit directly or indirectly from the '*Programming Language*' root class. These relationships can be described as "*is an instance of.*" For example, all subclasses show the relation "*is an instance of*" '*Programming Language*' such as '*High-level Compile Language*' "*is an instance of*" '*Compile Language*' "*is an instance of*" '*Programming Language*,' therefore '*High-level Compile Language*' also shows the relationship "*is an instance of*" '*Programming Language*'. This relationship is defined by the concept of inheritance where a subclass inherits the properties and attributes of its parent class and grandparent classes. Note that a subclass can have more than one parent class.

For example, the parent class '`Compiled_Language`' has a property '`Execution_Model`' with the attribute '`Complied`', the subclass '`High_Level_Compiled_Language`' and all the languages that are children of it will inherit the property '`Execution_Model`' with the attribute '`Complied`'. This can be translated into first-order logic as follows:

$$\forall x\, High\_Level\_Compiled\_Language(x) \Rightarrow Compiled\_Language(x) \Rightarrow Execution\_Model(x, "Compiled")$$

Where $'x'$ is the instance of a programming language (e.g. Java SE 23) and $' \Rightarrow '$ implies.

When exploring the Python frame we can see that one of its attributes is '`Instance_Of`' with the property '`Scripting_Language,`' this shows that Python is a subclass of the '`Scripting_Language`' class, therefore Python inherits all the properties and attributes from '`Scripting_Language`' which are '`Syntax: Easy_To_Use`', '`Execution_Model: Interpreted`', and '`Used_For: Software_Development`'. Additionally, the '`Syntax: Easy_To_Use`' is specific to '`Scripting_Language.`' On the other hand, '`Execution_Model: Interpreted`' and '`Used_For: Software_Development`' are inherited by '`Scripting_Language`' from '`Interprated_Language.`' Furthermore, '`Execution_Model: Interpreted`' is specific to '`Interprated_Language`' which inherits '`Used_For: Software_Development`' from '`Promming_Language.`' This can be translated into first-order logic as follows:

- $\forall x\, Python(x) \Rightarrow Scripting\_Language(x) \Rightarrow Syntax(x, Easy\_To\_Use)$
- $\forall x\, Python(x) \Rightarrow Scripting\_Language(x) \Rightarrow Interprated\_Language(x) \Rightarrow Execution\_Model(x, Interpreted)$
- $\forall x\, Python(x) \Rightarrow Scripting\_Language(x) \Rightarrow Interprated\_Language(x) \Rightarrow Promming\_Language \Rightarrow Used\_For(x, Software\_Development)$

Where 'x' is the instance of a programming language (e.g. Python 3.13) and '⇒' implies.
The rest of Python's properties and attributes are specific to it.

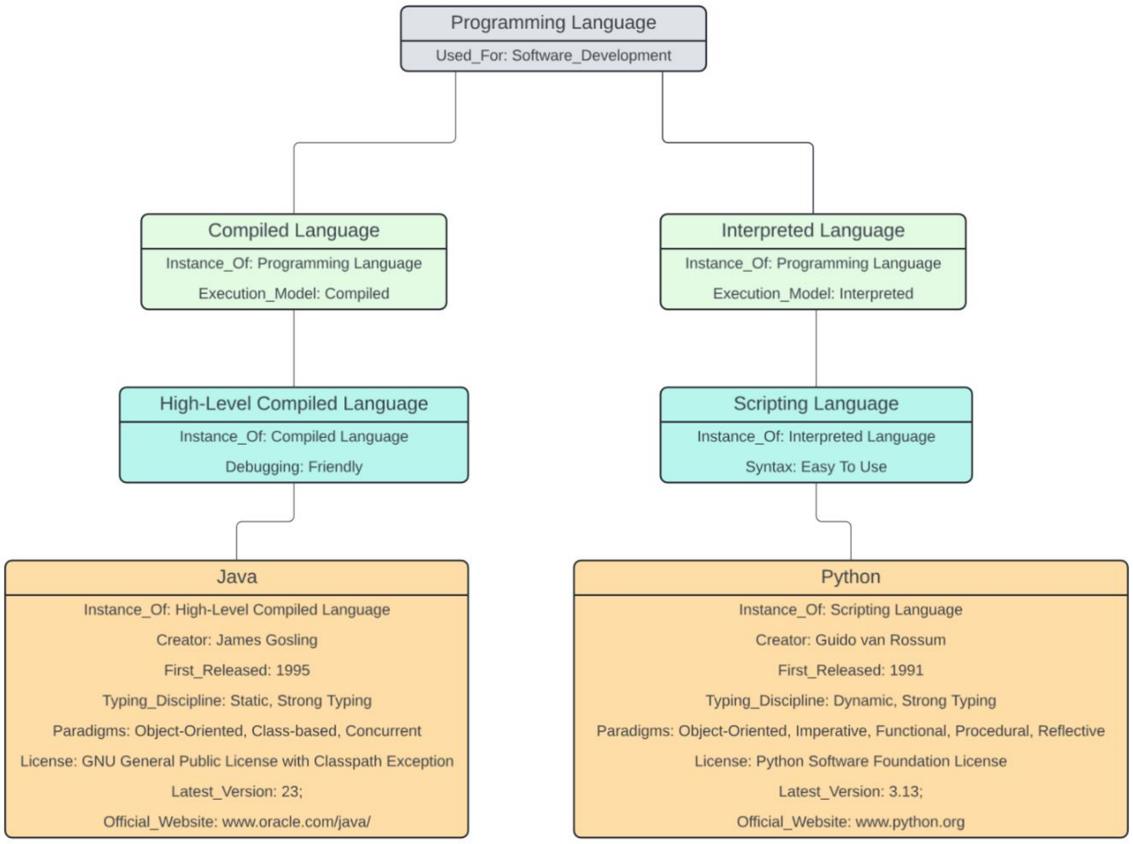
On a side note, in polymorphism, a subclass can modify (override) the attribute's value of a property inherited from a parent class. For example, a language could inherit '`Syntax`: `Easy_To_Use`' from '`Scripting_Language`' and modify the attribute '`Easy_To_Use`' to '`Hard_To_Use.`'

Frame and Hierarchical Taxonomy Interactions Visualization

This section illustrates visually, the interactions between the hierarchical taxonomy and the Java and Python frames.

Figure 4

Frame and Hierarchical Taxonomy Interactions (Java and Python)


Programming Language
Used_For: Software_Development
Compiled Language
Instance_Of: Programming Language
Execution_Model: Compiled
Interpreted Language
Instance_Of: Programming Language
Execution_Model: Interpreted
High-Level Compiled Language
Instance_Of: Compiled Language
Debugging: Friendly
Scripting Language
Instance_Of: Interpreted Language
Syntax: Easy To Use
Java
Instance_Of: High-Level Compiled Language
Creator: James Gosling
First_Released: 1995
Typing_Discipline: Static, Strong Typing
Paradigms: Object-Oriented, Class-based, Concurrent
License: GNU General Public License with Classpath Exception
Latest_Version: 23;
Official_Website: www.oracle.com/java/
Python
Instance_Of: Scripting Language
Creator: Guido van Rossum
First_Released: 1991
Typing_Discipline: Dynamic, Strong Typing
Paradigms: Object-Oriented, Imperative, Functional, Procedural, Reflective
License: Python Software Foundation License
Latest_Version: 3.13;
Official_Website: www.python.org

Note: The diagram illustrates the interactions between the hierarchical taxonomy and the Java and Python frames. Only the specific properties and attributes of the subclasses are listed in their node containers as the inherited properties and attributes can be listed in their parent class containers nodes. Data adapted from multiple sources: (Epözdemir, 2024; Foster, 2013; Gómez, n.d.; Peter Van Roy, 2008; Saxena, 2024; Startups, 2018; & Wikipedia contributors, 2024)

As shown in Figure 4, combining hierarchical taxonomic relationships and frames creates a powerful tool for representing knowledge. The hierarchical taxonomy illustrated the relationships between categories; for example, the '*Scripting Language*' category is a subcategory of the '*Interpreted Language*' which is a subcategory of the '*Programming Language*' category making the 'Scripting Language' a sub-subcategory of the root category '*Programming Language*' which represents the domain. Additionally, the implementation of frames into the diagram shows the entities' properties and attributes and how they get inherited from another category. For example, Python's specific properties and attributes are listed in its node containers, and its inherited properties and attributes are listed in its parent, grandparent, and great-grandparent classes node containers. This creates a robust representation of knowledge that provides depth and clarity allowing users to navigate complex relationships effortlessly.

References

Colorado State University Global. (n.d.). *Module 4: Knowledge Representation* [Interactive lecture]. Canvas. Retrieved November 1, 2024, from https://csuglobal.instructure.com/courses/100844/pages/4-dot-2-frames?module_item_id=5183634

Epözdemir, J. (2024, April 10). Programming Language Categories - Jiyan Epözdemir - Medium. *Medium*. https://medium.com/@jepozdemir/programming-language-categories-6b786d70e8f7

Foster, D. (2013, February 20). Visual Guide to Programming Language Properties. *DaFoster*. https://dafoster.net/articles/2013/02/20/visual-guide-to-programming-language-properties/

Gómez, R. (n.d.). *Alphabetical List of Programming Languages • programminglanguages.info*. https://programminglanguages.info/languages/

Knight, M. (2021, March 12). *What Is Taxonomy?* Dataversity. https://www.dataversity.net/what-is-taxonomy/

Novak, J., Krajnc, A., & Žontar, R. (2010, May 1).*Taxonomy of static code analysis tools*. IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/5533417

Peter Van Roy. (2008). *The principal programming paradigms*. https://webperso.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf

Saxena, C. (2024, October 17). *Top Programming Languages 2025: By Type and Comparison*. ISHIR | Software Development India. https://www.ishir.com/blog/36749/top-75-programming-languages-in-2021-comparison-and-by-type.htm

Startups, A. (2018, June 20). Choosing the Right Programming Language for Your Startup. *Medium*. https://medium.com/aws-activate-startup-blog/choosing-the-right-programming-language-for-your-startup-b454be3ed5e2

Wikipedia contributors. (2024, November 3). *List of programming languages by type*. Wikipedia. https://en.wikipedia.org/wiki/List_of_programming_languages_by_type