A.V.V.M SRI PUSHPAM COLLEGE (AUTONOMOUS),
POONDI – 613503, THANJAVUR (DIST)

Department of Computer Science

EDUCATION ONTOLOGY DEVELOPMENT

User Guide for Protégé Tool

Education Ontology Development

Definition of Ontology

- In 1993, Gruber originally defined the notion of ontology as an "**explicit specification of a conceptualization**".

Concept of Ontology

- Ontology is the branch of philosophy that studies concepts such as existence, being, becoming, and reality.

Importance of the study of Ontology in Education

- The ontology or an appropriate version of it can be used to guide students to understand the organization of their own learning and to self- assess their own progress. The ontologies are created by sets of people with expertise in content, teaching, psychology, and measurement.

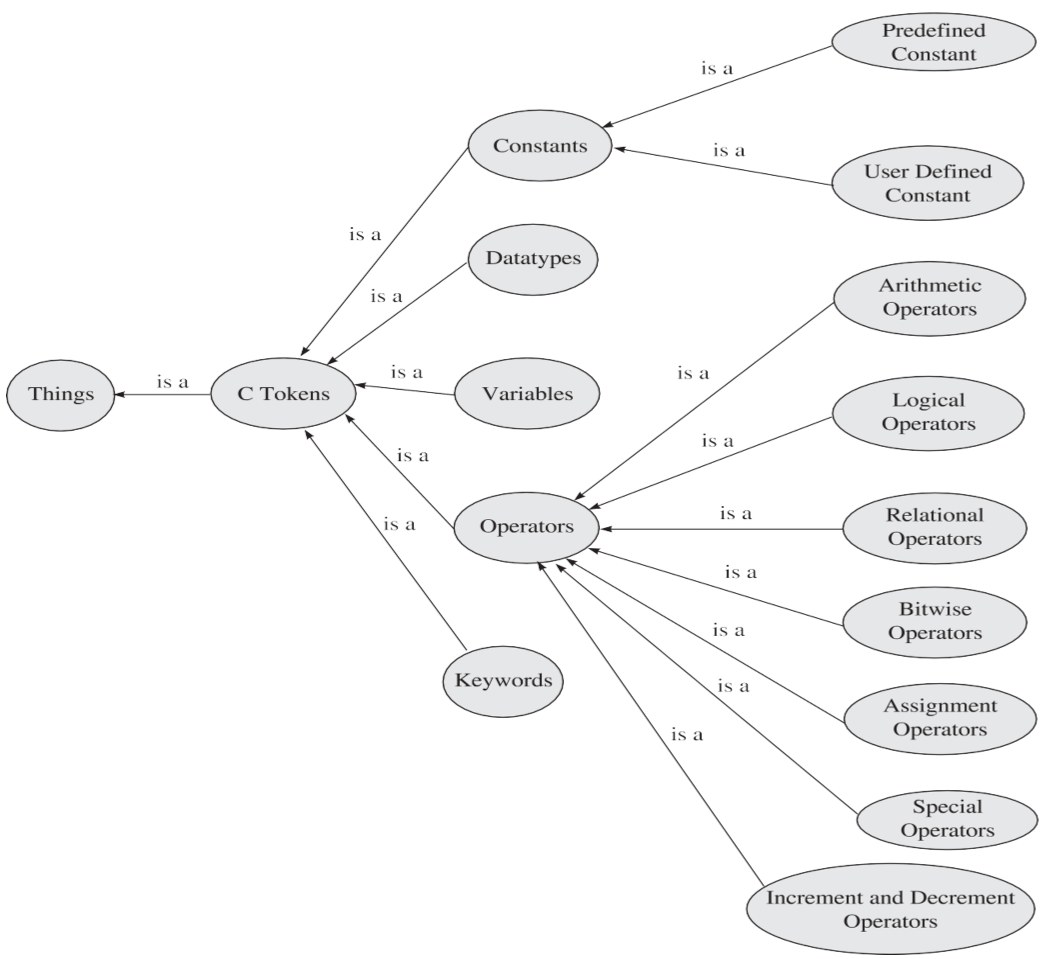
PROTÉGÉ TOOL

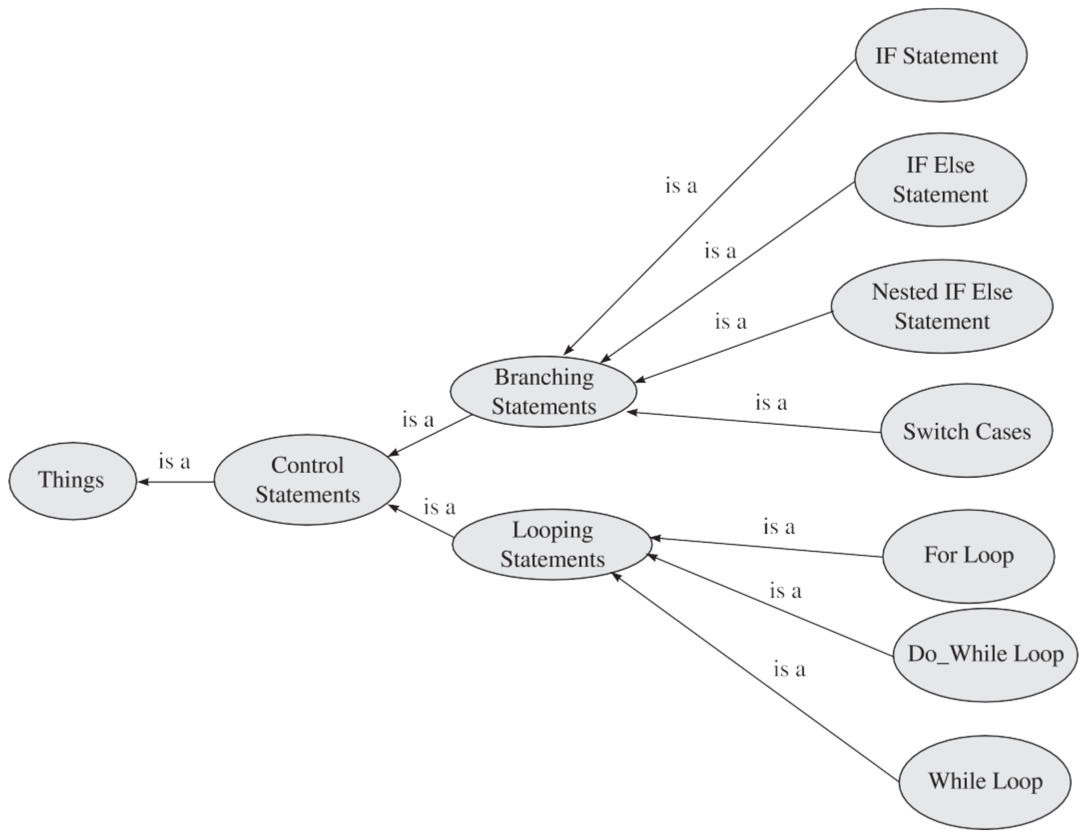
- Protégé is an ontology and knowledge base editor produced by Stanford University.
- Protégé is a tool that enables the construction of domain ontologies, customized data entry forms to enter data.
- Protégé allows the definition of classes, class hierarchies, variables, variable-value restrictions, and the relationships between classes and the properties of these relationships.
- Protégé comes with visualization packages such as OntoViz; all of these help the user visualize ontologies with the help of diagrams.

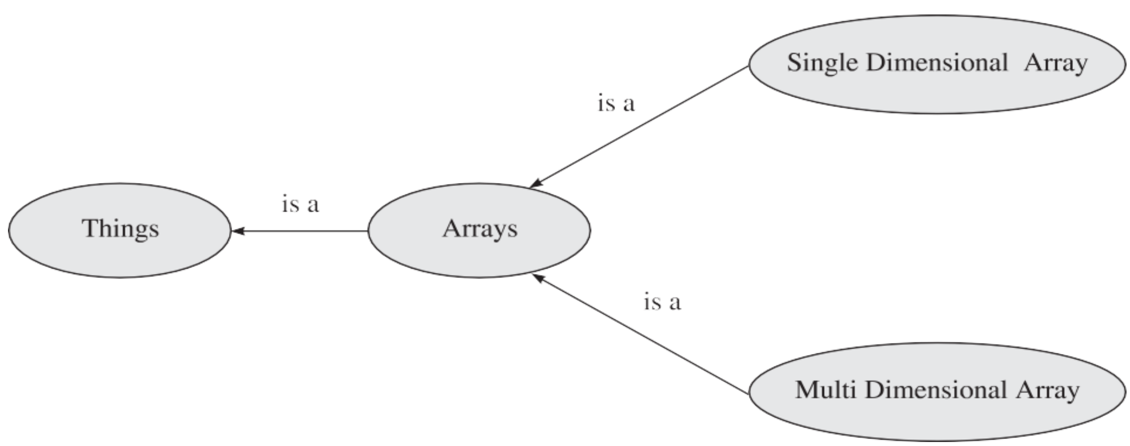
CASE STUDY 1: TEACHING ONTOLOGY WITH C PROGRAMMING LANGUAGE

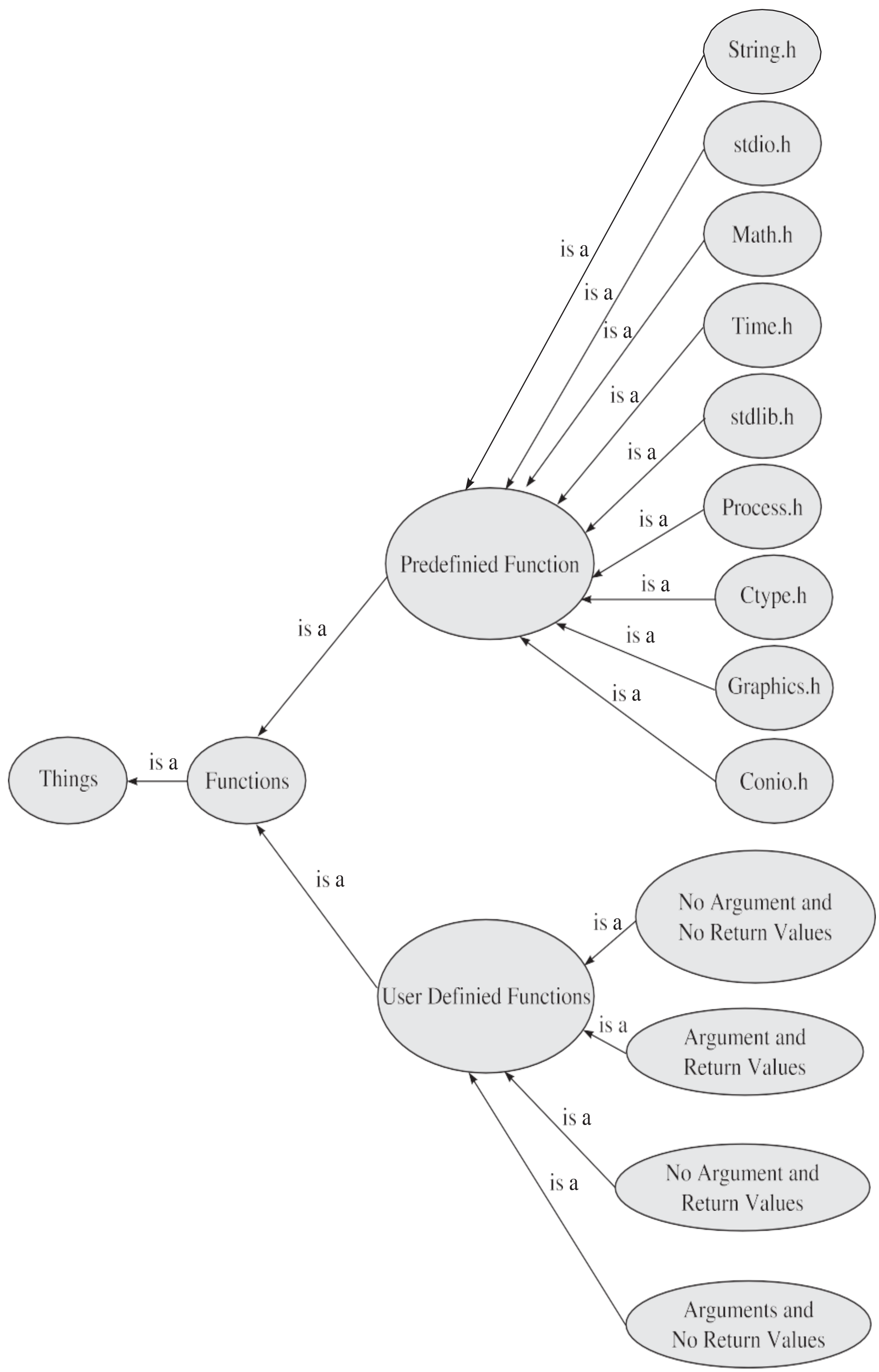
This case study demonstrates how to create ontology for the domain of *"**C Programming Language**"*.

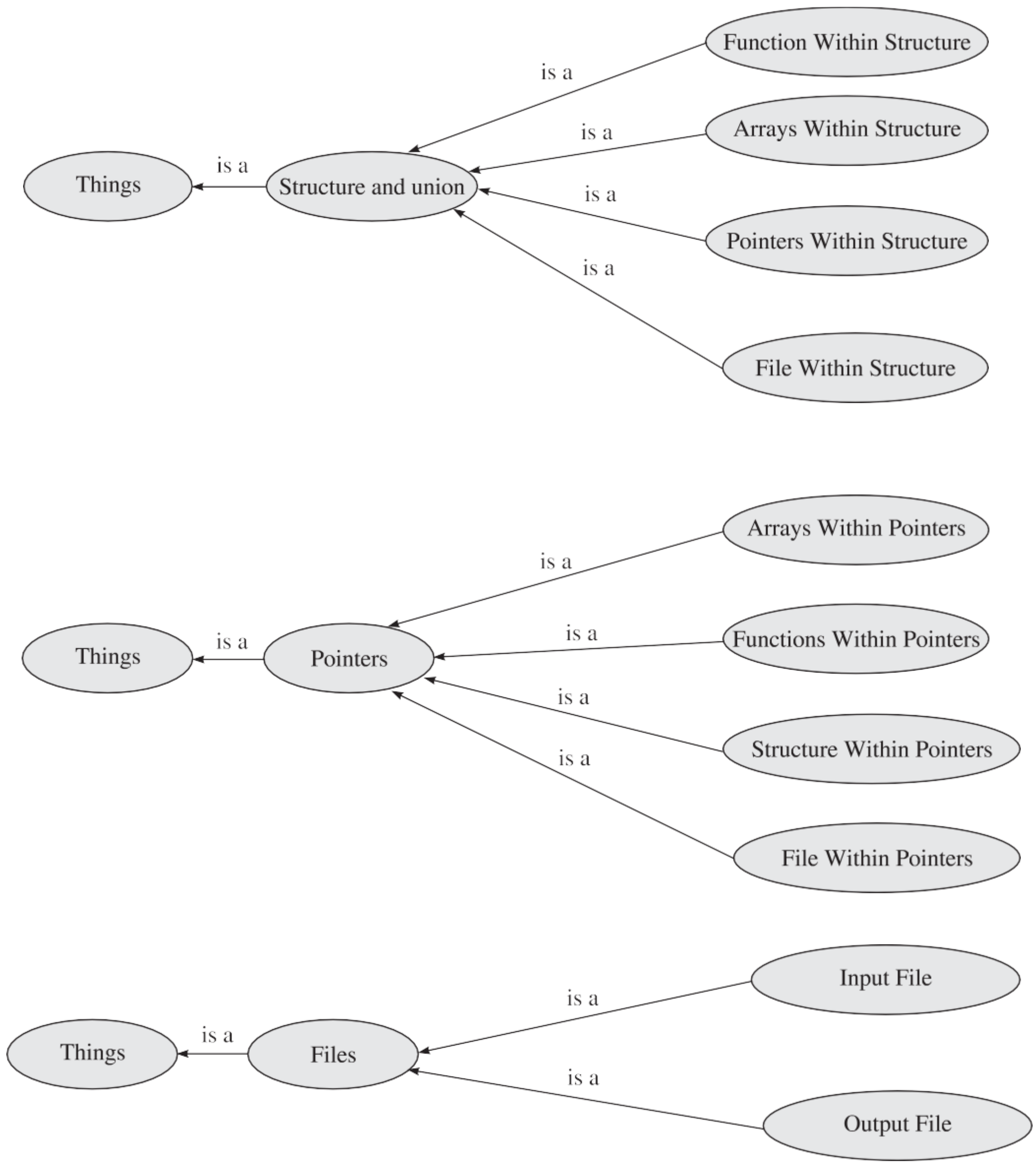
C is a robust language whose rich set of built-in functions and operators can be used to write any complex programs. Programs written in C are efficient, fast and highly portable. Its strength lies in its built-in functions. It is well-suited for structured programming. It has the ability to extend itself. The specific characteristics are low-level access to computer memory by converting machine addresses to typed pointers, functions, structures, unions, I/O string manipulation, files and mathematical functions. Figure describes this scenario through a class diagram.




IF Statement
IF Else Statement
Nested IF Else Statement
Switch Cases
Branching Statements
is a
Things
Control Statements
Looping Statements
For Loop
Do_While Loop
While Loop


Single Dimensional Array
is a
Things
Arrays
Multi Dimensional Array


String.h
stdio.h
Math.h
Time.h
stdlib.h
Process.h
Ctype.h
Graphics.h
Conio.h
Predefinied Function
Things
Functions
User Definied Functions
No Argument and
No Return Values
Argument and
Return Values
No Argument and
Return Values
Arguments and
No Return Values
is a



Scenario of C programming language through a class diagram.

Identifying the Activities to Build C Teaching Ontology

To build a C teaching ontology, the following activities are required:

Activity 1: Identification and Creation of Classes and Subclasses

Activity 2: Identification and Creation of Object Properties

Activity 3: Identification and Creation of Datatype Properties

Activity 4: Defining Disjoint Classes

Activity 5: Specify Property Characteristics

Activity 6: Assign Domain and Range

Activity 7: Identify and Create Instance or Individual

Activity 1: Identification and creation of classes and subclasses

Identified classes and subclasses are shown in Exhibit 1.1.

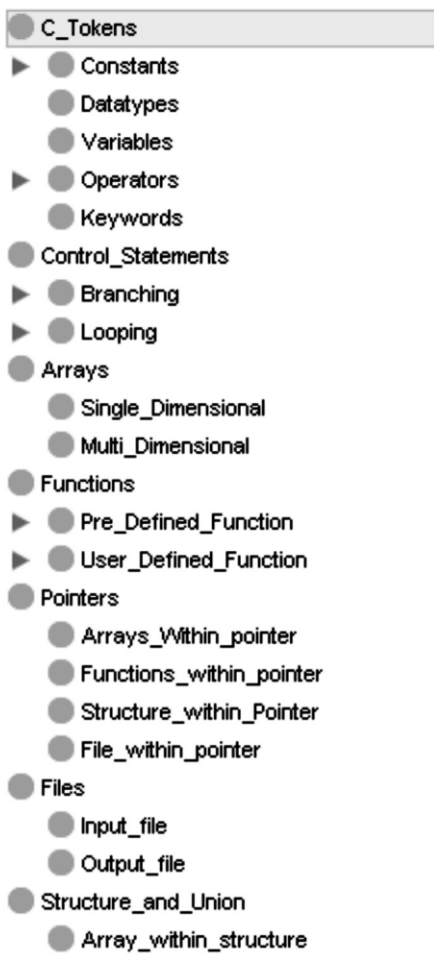


Exhibit 1.1 Identified classes and subclasses.

Activity 2: Identification and creation of object properties

Identified object properties are as follows:

- ☐ hasPart
- ☐ isPartOf
- ☐ hasType
- ☐ isTypeOf
- ☐ has Array
- ☐ isArrayOf

They are shown in Exhibit 1.2.

Exhibit 1.2 Identified object properties.

Activity 3: Identification and creation of datatype properties

Identified datatype properties are as follows:

- ☐ hasSubscript
- ☐ isSubscriptOf

They are shown in Exhibit 1.3.

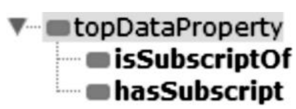


Exhibit 1.3 Identified datatype properties.

Activity 4: Defining disjoint classes

In the class definition, the classes "*pointers*" and "*Structures and Unions*" should be defined as *Disjoint Classes* and is shown in Figure 1.21.

Activity 5: Specify property characteristics

Ontology permits the meaning of properties to be improved through the use of property characteristics. Following are the different property characteristics:

- ☐ Functional
- ☐ Inverse Functional
- ☐ Transitive
- ☐ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

Functional: The "*hasArray*" property should be assigned *functional* characteristic. It is shown in Figure 1.22.

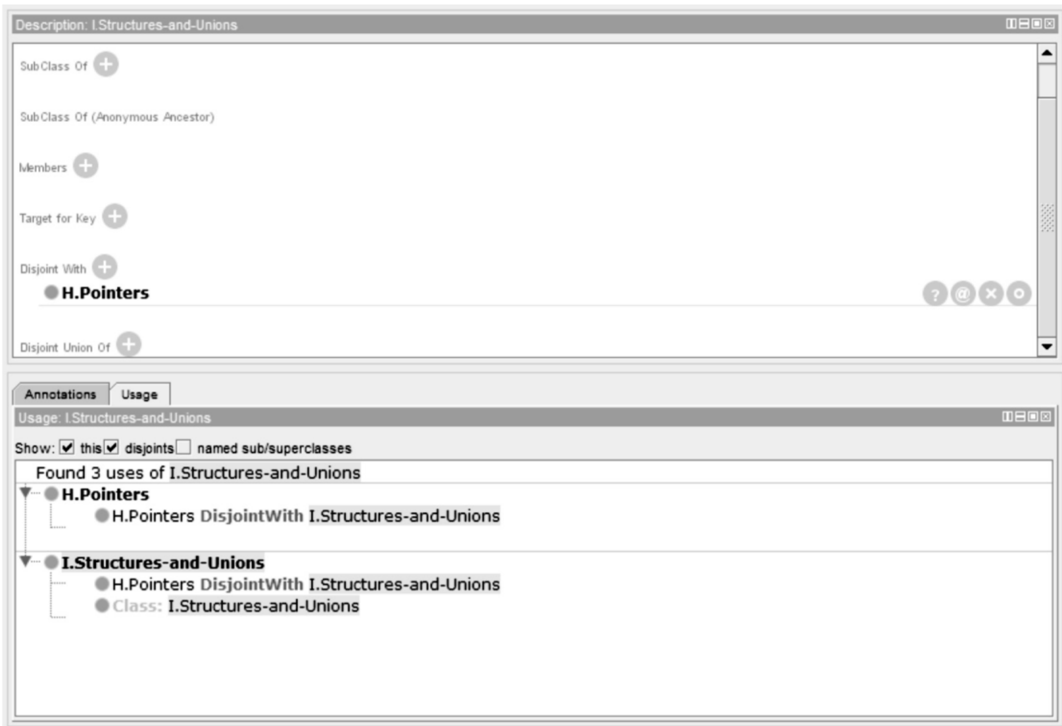


Figure 1.21 Disjoint Classes—*Pointers*, *Structures and Unions.*

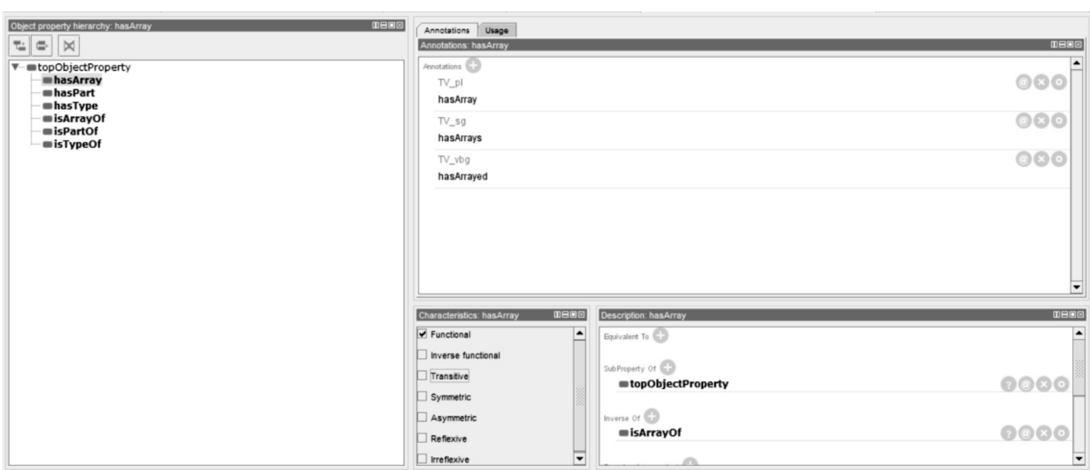


Figure 1.22 Functional characteristic—*hasArray.*

Inverse functional: The "*hasPart*" property should be assigned *Inverse functional* characteristic. It is shown in Figure 1.23.

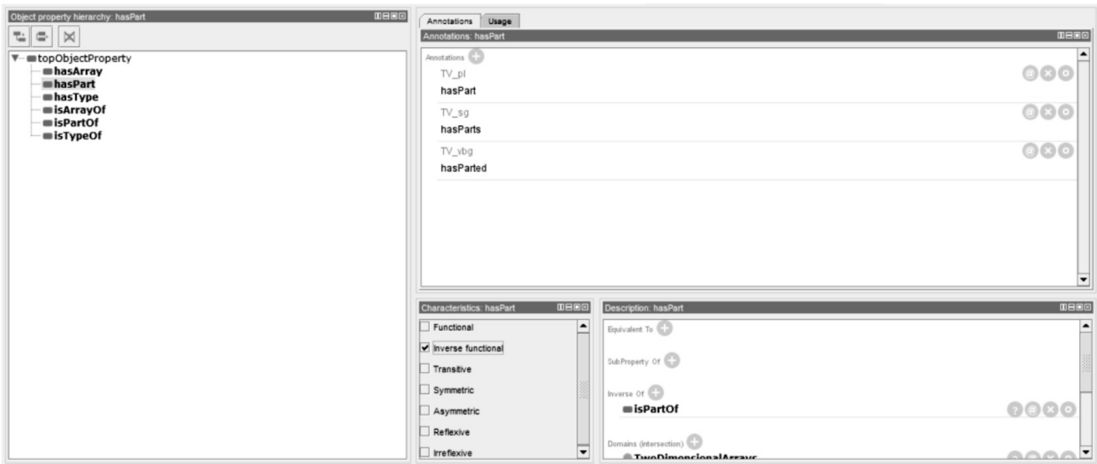


Figure 1.23 Inverse functional characteristic—*hasPart.*

Transitive: The "*hasType*"property should be assigned *Transitive* characteristic. It is shown in Figure 1.24.

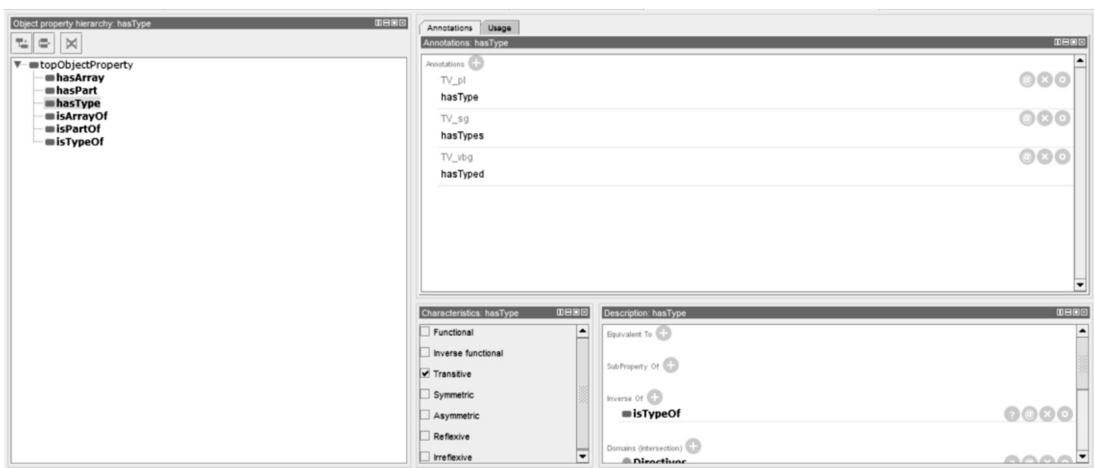


Figure 1.24 Transitive characteristic—*hasType.*

Symmetric: The "*isArrayOf*" property should be assigned *Symmetric* characteristic. It is shown in Figure 1.25.

Asymmetric: The "*isPartOf*" property should be assigned *Asymmetric* characteristic. It is shown in Figure 1.26.

Reflective: The "i*sTypeOf*"property should be assigned *Reflective* characteristic. It is shown in Figure 1.27.

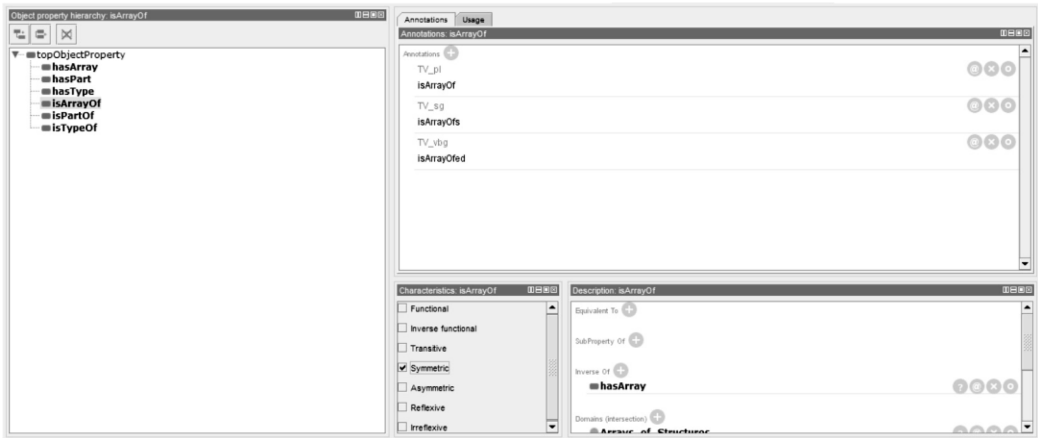


Figure 1.25 Symmetric characteristic—is*ArrayOf.*

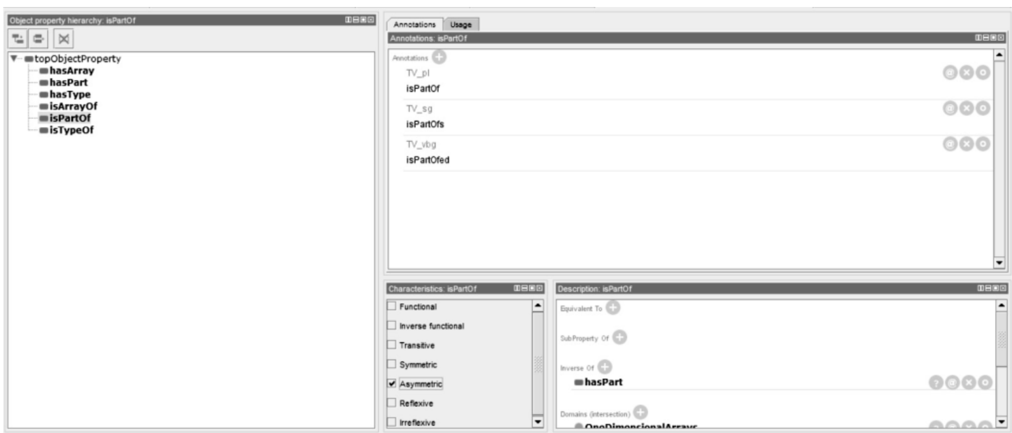


Figure 1.26 Asymmetric characteristic—*isPartOf.*

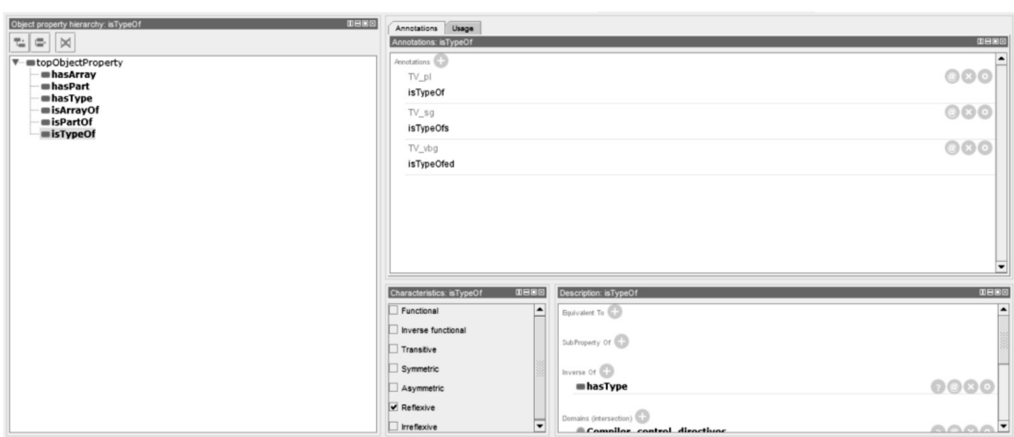


Figure 1.27 Reflective characteristic—is*TypeOf.*

Activity 6: Assign domain and range

Properties may have specified *domain* and *range*. Properties link individuals to individuals (from the *domain* and the *range*). For example, in this domain ontology, the property *isPartOf* would probably link individuals belonging to the class of *OneDimensionalArrays* to individuals belonging to the class of *TwoDimensionalArrays*. In this case, the domain of the property '*isPartOf* ' is *OneDimensionalArrays* (Figure 1.28) and the range is TwoDimensionalArrays (Figure 1.29).

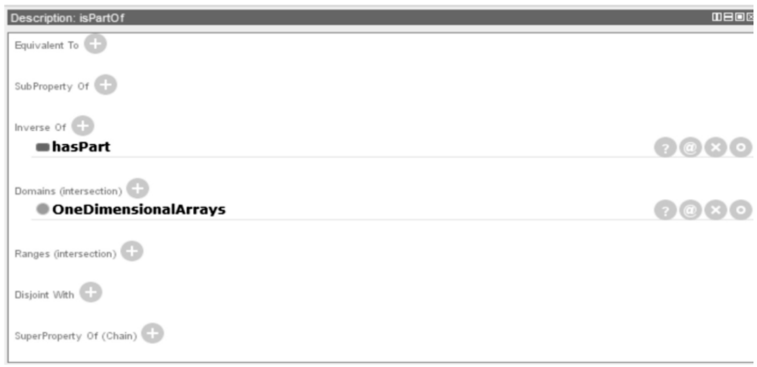


Figure 1.28 Property domain view—*OneDimensionalArrays*.

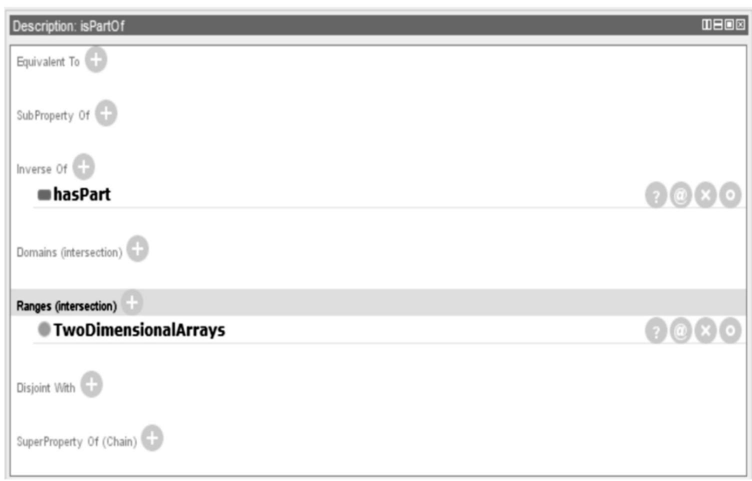


Figure 1.29 Property range view—*TwoDimensionalArrays.*

Activity 7: Identify and create instance or individual

Identified individuals or instances are shown in Table 1.7.

Table 1.7 Identified Individuals

Classes	Instances
Constants	PI constant
Data types	Integer Data type, Float Data type, Character Data type
Logical Operators	AND Operators, OR Operators, Not Operators
Structure	Defining a structure Initializing a structure
Pointers	Defining a pointer Initializing a pointer
Arrays	Defining an Array Initializing an Array