



Jhanvi Parekh
60009210033
D11

Experiment 9

Aim:

Implement a rule based expert system using Protégé 5.5.

Theory:

The strength of an Expert System derives from its knowledge base - an organized collection of facts and heuristics about the system's domain. An ES is built in a process known as knowledge engineering, during which knowledge about the domain is acquired from human experts and other sources by knowledge engineers.

The accumulation of knowledge in knowledge bases, from which conclusions are to be drawn by the inference engine, is the hallmark of an expert system.

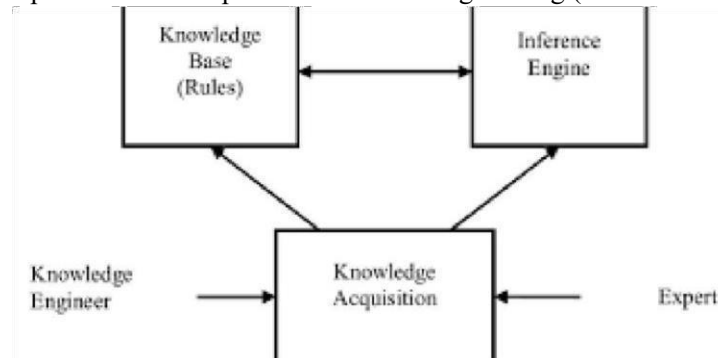
Knowledge Representation and the Knowledge Base

The knowledge base of an ES contains both factual and heuristic knowledge. Knowledge representation is the method used to organize the knowledge in the knowledge base.

Knowledge bases must represent notions as actions to be taken under circumstances, causality, time, dependencies, goals, and other higher-level concepts.



Department of Computer Science and Engineering (Data Science)



Several methods of knowledge representation can be drawn upon. Two of these methods include:

1. Frame-based systems - are employed for building very powerful ESs. A frame specifies the attributes of a complex object and frames for various object types have specified relationships.
2. Production rules - are the most common method of knowledge representation used in business.

Rule-based expert systems are expert systems in which the knowledge is represented by production rules.

A production rule, or simply a rule, consists of an IF part (a condition or premise) and a THEN part (an action or conclusion). IF condition THEN action (conclusion).

The explanation facility explains how the system arrived at the recommendation. Depending on the tool used to implement the expert system, the explanation may be either in a natural language or simply a listing of rule numbers.

Inference Engine

The inference engine:

1. Combines the facts of a specific case with the knowledge contained in the knowledge base to come up with a recommendation. In a rule-based expert system, the inference engine controls the order in which production rules are applied and resolves conflicts if more than



Department of Computer Science and Engineering (Data Science)

one rule is applicable at a given time. This is what A reasoning amounts to in rule-based systems.

2. Directs the user interface to query the user for any information it needs for further inferencing.

The facts of the given case are entered into the working memory, which acts as a blackboard, accumulating the knowledge about the case at hand. The inference engine repeatedly applies the rules to the working memory, adding new information (obtained from the rules conclusions) to it, until a goal state is produced or confirmed.

Lab Assignment to do:

1. Implement access control use case with Inquirer, Data, Hospital and Patients as Main Classes.
2. Design appropriate Ontology and test it using Hermit Reasoner.
3. Execute Drool inference engine and observe the changes in Onto Graph.

Working:

1. Creating an ontology

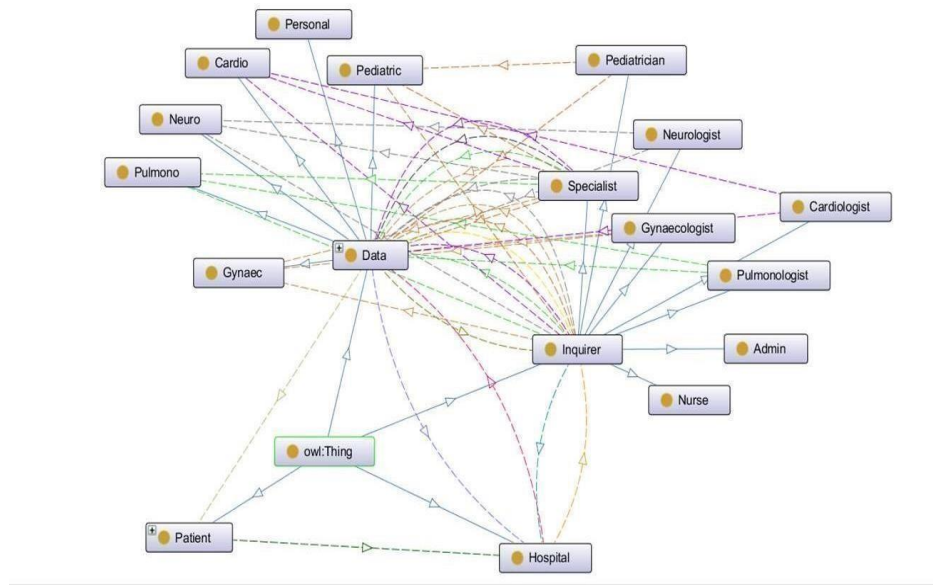
Metrics	
Axiom	282
Logical axiom count	214
Declaration axioms count	66
Class count	20
Object property count	14
Data property count	16
Individual count	17
Annotation Property count	3

Class axioms	
SubClassOf	16
EquivalentClasses	3
DisjointClasses	3
SDI count	3
HasPart-ODI Count	3

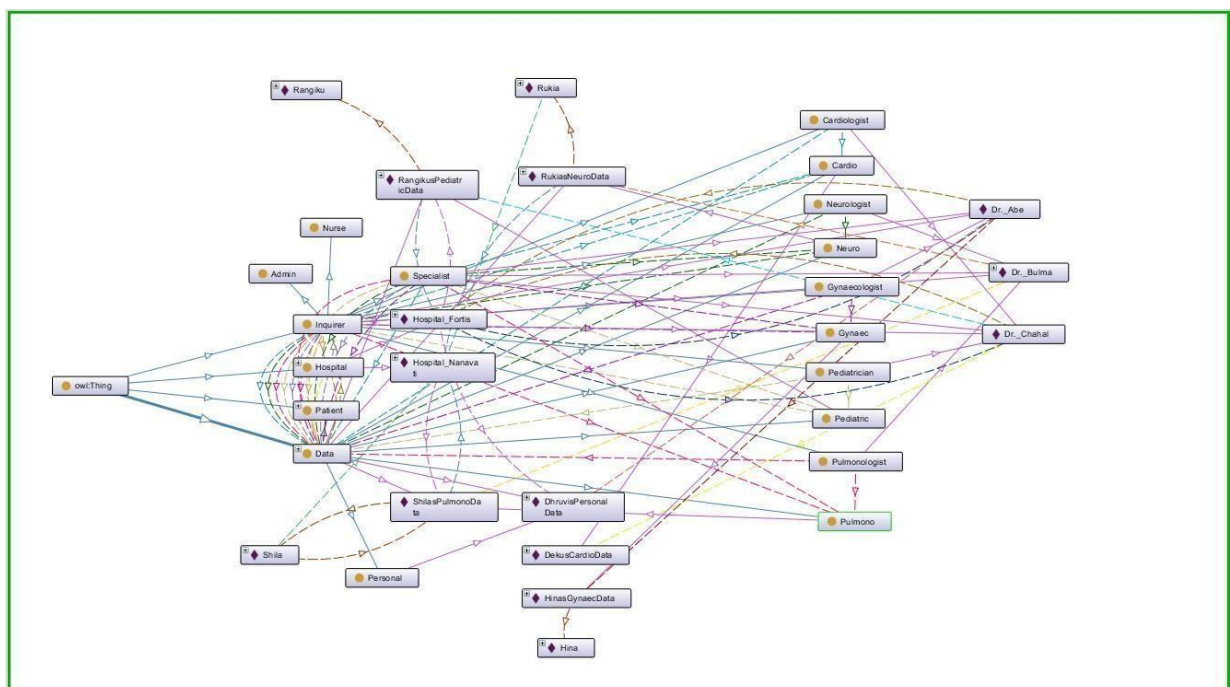
Object property axioms	
SubObjectPropertyOf	3
EquivalentObjectProperties	3
InverseObjectProperties	3
DisjointObjectProperties	3
FunctionalObjectProperty	3
InverseFunctionalObjectProperty	3
TransitiveObjectProperty	3

Department of Computer Science and Engineering (Data Science)

2. Under Entities, creating Classes for Data, Hospital, Inquirer, Patient



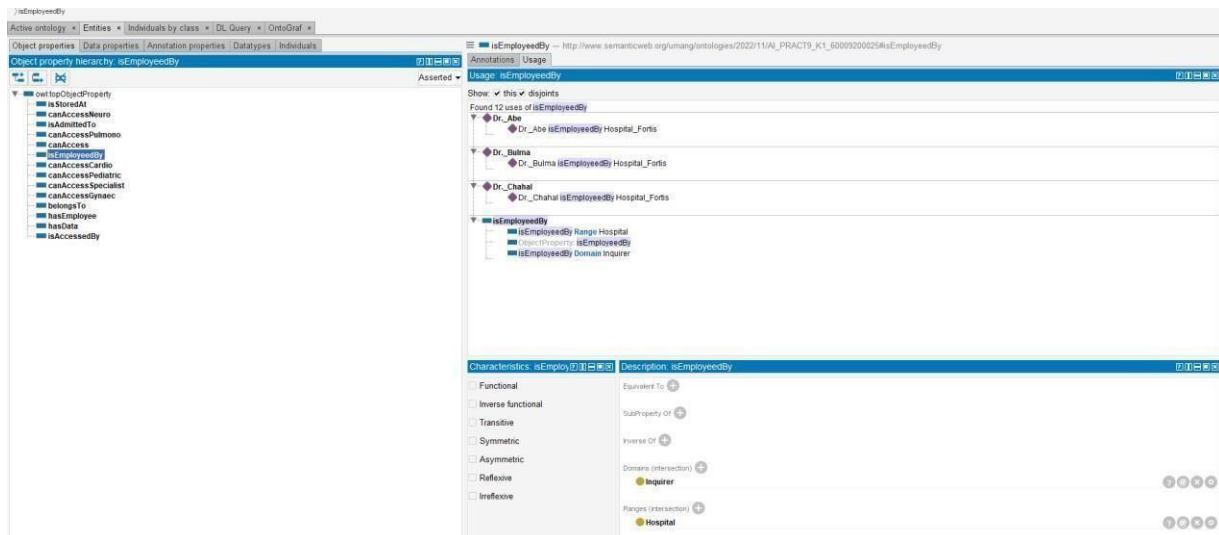
3. Creating an Onto Graph



4. Defining object properties

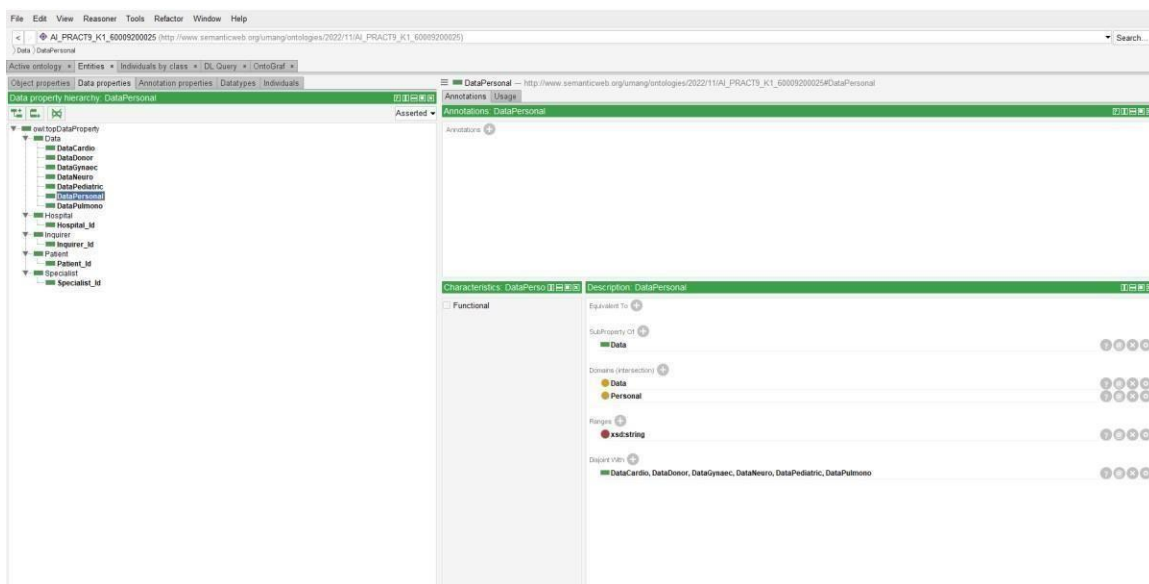


Department of Computer Science and Engineering (Data Science)



Different Object Properties are: belongsTo, canAccess, canAccess<Specific>, hasData, isAccessedBy, isAdmittedTo, isEmployedBy, isStoredAt.

5. Defining different Data Properties



Different Data Properties are: Data, Data<Specific>, Hospital_Id, Inquirer_Id, Patient_Id, Specialist_Id.

6. Defining Individuals



Department of Computer Science and Engineering (Data Science)

Active ontology: Entires: Individuals by class: DL Query: OntoGraf

Object properties: Data properties: Annotation properties: Datatypes: Individuals

Individuals: ShilasPulmonoData

Annotations: Usage

Usage: ShilasPulmonoData

Show: ✓ this ✓ different

Found 20 uses of ShilasPulmonoData

- Dr. Bulma
- Dr. Bulma canAccessPulmono ShilasPulmonoData
- Hospital_Nanavati
- Hospital_Nanavati hasData ShilasPulmonoData
- Shila
- Shila belongsTo ShilasPulmonoData
- ShilasPulmonoData
- ShilasPulmonoData Data "PulmonoData1"^^xsd:string
- ShilasPulmonoData Type Data

Description: ShilasPulmonoData

Types: Data, Pulmono

Some individual As: Different individuals

Property assertions: ShilasPulmonoData

Object property assertions: is StoredAt Hospital_Nanavati, belongsTo Shila

Data property assertions: Data "PulmonoData1"^^xsd:string, DataPulmono "PulmonoData1"^^xsd:string

Negative object property assertions: Negative data property assertions:

7. Summary – Individuals by Classes
a. Individuals by Class: Data

Active ontology: Entires: Individuals by class: DL Query: OntoGraf

Class hierarchy: Cardio

Asserted

Usage: ShilasPulmonoData

Show: ✓ this ✓ different

Found 20 uses of ShilasPulmonoData

- Dr. Bulma
- Dr. Bulma canAccessPulmono ShilasPulmonoData
- Hospital_Nanavati
- Hospital_Nanavati hasData ShilasPulmonoData
- Shila
- Shila belongsTo ShilasPulmonoData
- ShilasPulmonoData
- ShilasPulmonoData Data "PulmonoData1"^^xsd:string
- ShilasPulmonoData Type Data
- ShilasPulmonoData DataPulmono "PulmonoData1"^^xsd:string
- ShilasPulmonoData Type Pulmono
- ShilasPulmonoData belongsTo Shila
- ShilasPulmonoData isStoredAt Hospital_Nanavati
- Individual: ShilasPulmonoData

Description: ShilasPulmonoData

Types: Data, Pulmono

Some individual As: Different individuals

Property assertions: ShilasPulmonoData

Object property assertions: is StoredAt Hospital_Nanavati, belongsTo Shila

Data property assertions: Data "PulmonoData1"^^xsd:string, DataPulmono "PulmonoData1"^^xsd:string

Negative object property assertions: Negative data property assertions:



Department of Computer Science and Engineering (Data Science)

b. Individuals by Class:
Hospital

The screenshot shows the Protege software interface with the 'Hospital' class selected in the class hierarchy. The left pane shows the hierarchy with 'Hospital' under 'Patient'. The right pane shows the 'Usage' tab for 'Hospital', listing 22 uses. The bottom pane shows the 'Direct instances' for 'Hospital', listing 'Hospital_Fortis' and 'Hospital_Nanavati'. The 'Property assertions' pane on the right shows assertions for 'Hospital'.

c. Individuals by Class
Inquirer

The screenshot shows the Protege software interface with the 'Inquirer' class selected in the class hierarchy. The left pane shows the hierarchy with 'Inquirer' under 'Cardiologist'. The right pane shows the 'Usage' tab for 'Inquirer', listing 25 uses. The bottom pane shows the 'Direct instances' for 'Inquirer', listing 'Dr. Chahal'. The 'Property assertions' pane on the right shows assertions for 'Inquirer'.



Department of Computer Science and Engineering (Data Science)
d. Individuals by Class Patient

The screenshot displays the Protégé ontology editor interface. On the left, the 'Class Hierarchy' pane shows a tree structure with 'Patient' as the root class. Under 'Patient', there are several subclasses: 'Cardio', 'Gynaec', 'Neuro', 'Pediatric', 'Personal', 'Pulmono', 'Hospital', 'Inquire', 'Admin', 'Cardiologist', 'Gynaecologist', 'Neurologist', 'Nurse', 'Pediatrician', 'Pulmonologist', and 'Specialist'. The 'Direct instances: Shila' pane shows a list of instances: 'Devi', 'Dhruvi', 'Hina', 'Rangika', 'Rukia', and 'Shila'. The 'Description: Shila' pane shows the URI 'http://www.semanticweb.org/pulmonangiontologies/2022/11/SH_PRACT9_K1_6009200025#Shila'. The 'Property assertions: Shila' pane shows several assertions: 'Shila belongsTo ShilasPulmonoData', 'Shila Patient_Id "PulmonoPatient1"^^xsd:string', 'Shila Type Patient', 'Shila Patient "PulmonoPatient1"^^xsd:string', and 'Shila isAdmittedTo Hospital_Nanavati'. The 'Usage: Shila' pane shows 14 uses of 'Shila'.

8. Running Query



Department of Computer Science and Engineering (Data Science)

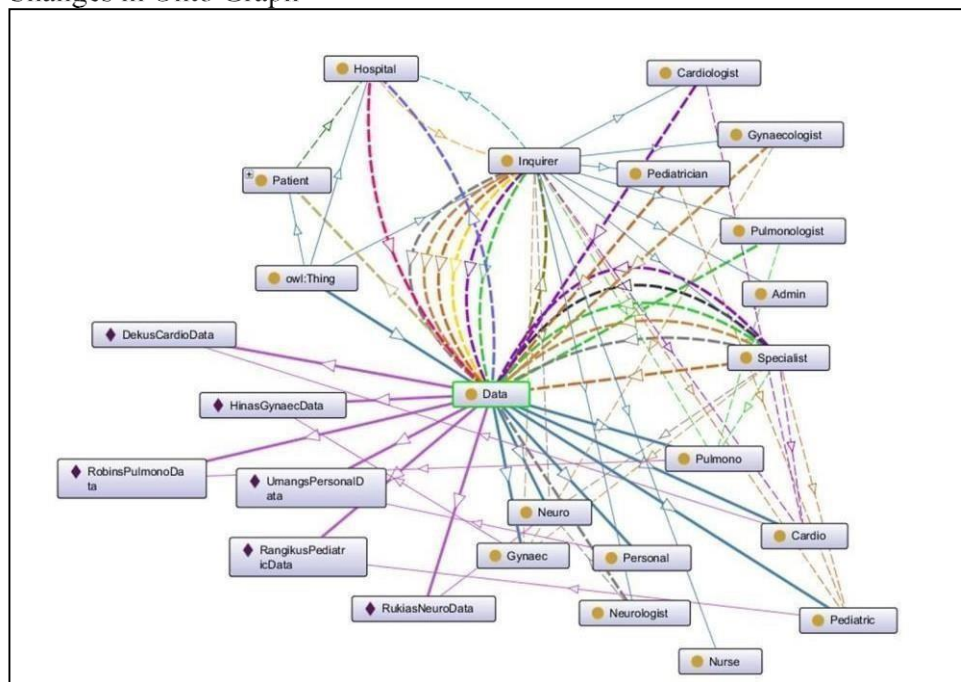
Active ontology | Entities | Individuals by class | DL Query | OntoGraf | SWRLTab

General	Name	Rule
<input checked="" type="checkbox"/>	Specialist(?a) ^ Regular(?y) ^ belongsTo(?a, ?y) ^ isEmployedBy(?a, ?H) ^ Hospital(?H) -> isAccessedBy(?a, ?x)	

Control | Rules | Asserted Axioms | Inferred Axioms | OWL 2 RL

OWL axioms successfully transferred to rule engine.
Number of SWRL rules exported to rule engine: 1
Number of OWL class declarations exported to rule engine: 20
Number of OWL individual declarations exported to rule engine: 17
Number of OWL object property declarations exported to rule engine: 14
Number of OWL data property declarations exported to rule engine: 16
Total number of OWL axioms exported to rule engine: 249
The transfer took 19 millisecond(s).
Press the 'Run Drools' button to run the rule engine.
Successful execution of rule engine.
Number of inferred axioms: 247.
The process took 338 millisecond(s).
Look at the 'Inferred Axioms' tab to see the inferred axioms.
Press the 'Drools -> OWL' button to translate the inferred axioms to OWL knowledge.
Successfully transferred inferred axioms to OWL model.
The process took 32 millisecond(s).

9. Changes in Onto Graph



Thus, we have successfully implemented a rule based expert system for a hospital using Protégé 5.5