



Department of Computer Science and Engineering (Data Science)

Subject: Artificial Intelligence (DJ19DSC502)

AY: 2022 - 23

Experiment 9

(Knowledge Representation and Reasoning)

Name: Umang Kirit Lodaya

Sap ID: 60009200032

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.

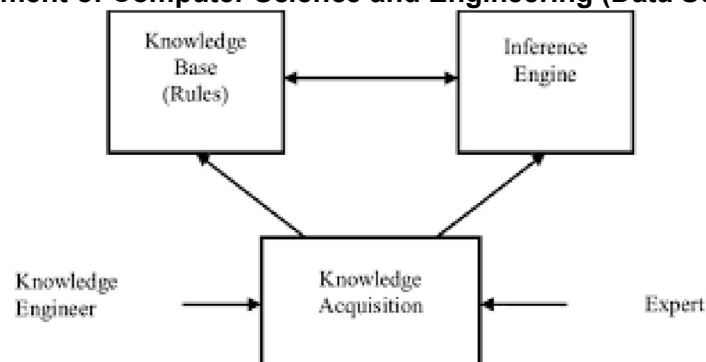
Knowledge Representation and the Knowledge Base

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.



Department of Computer Science and Engineering (Data Science)

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 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.



Department of Computer Science and Engineering (Data Science)

Working:

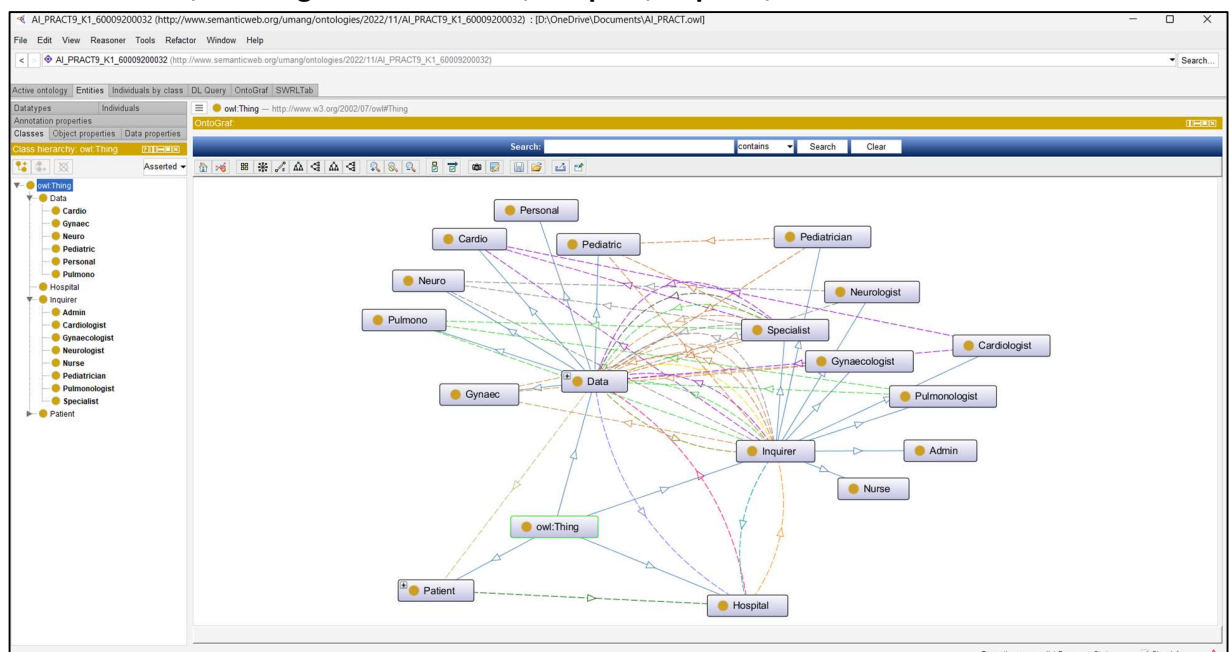
1. Creating an Ontology

Metrics	
Axiom	281
Logical axiom count	214
Declaration axioms count	67
Class count	20
Object property count	14
Data property count	16
Individual count	17
Annotation Property count	3

Class axioms	
SubClassOf	16
EquivalentClasses	0
DisjointClasses	0
GCI count	0
Hidden GCI Count	0

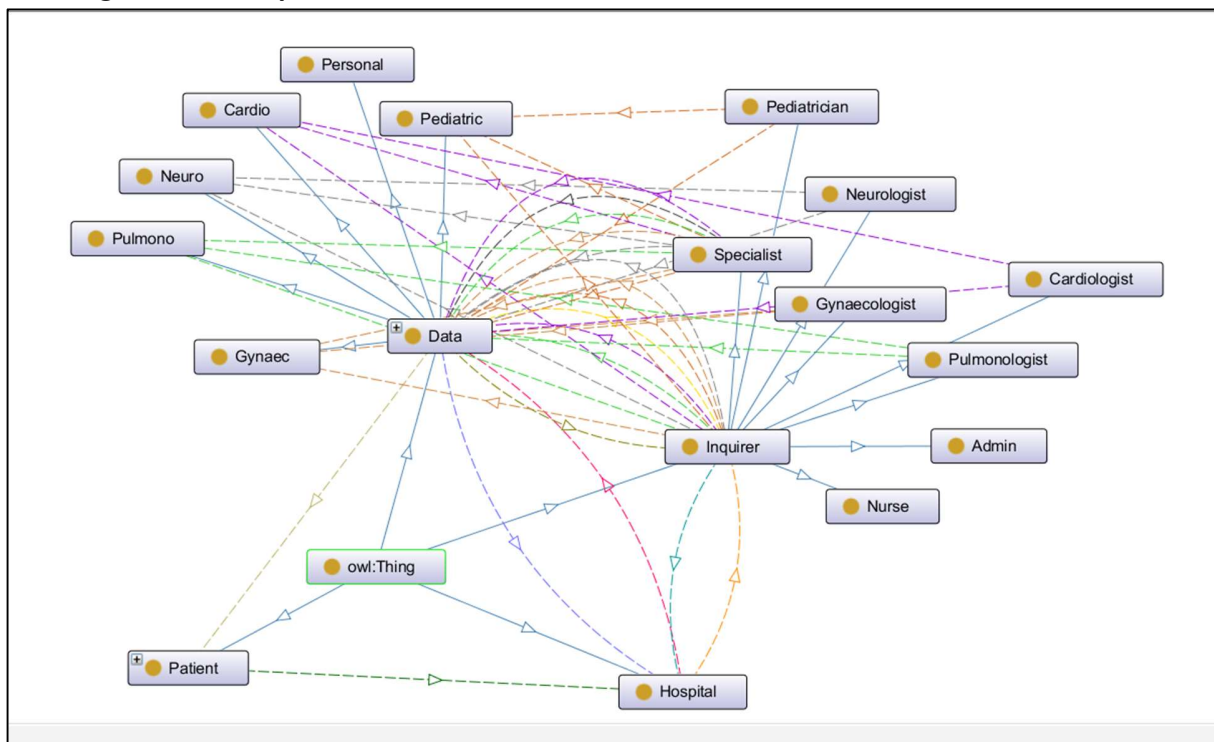
Object property axioms	
SubObjectPropertyOf	0
EquivalentObjectProperties	0
InverseObjectProperties	0
DisjointObjectProperties	0
FunctionalObjectProperty	0
InverseFunctionalObjectProperty	0

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

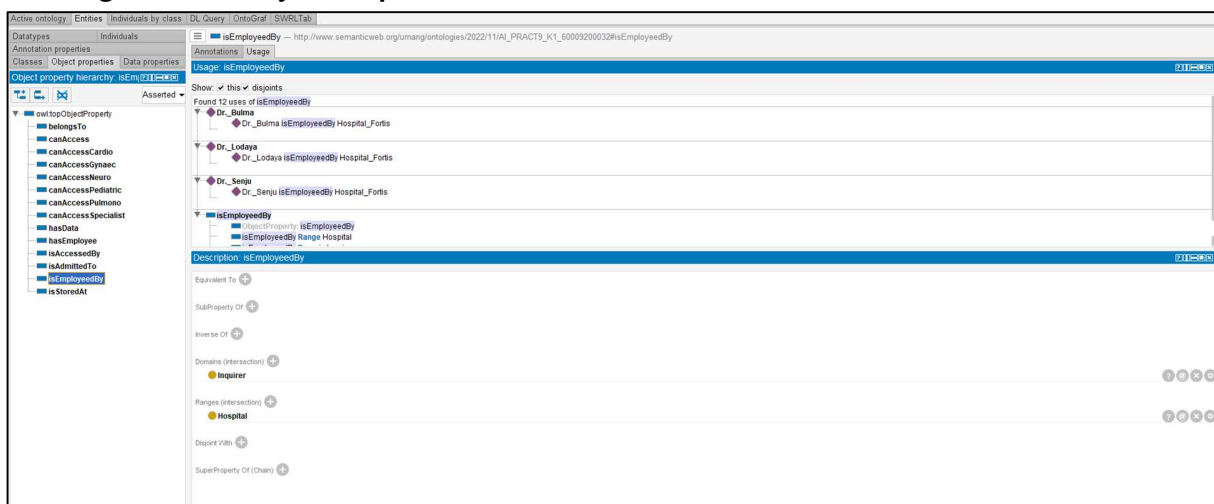


Department of Computer Science and Engineering (Data Science)

3. Creating an Onto Graph



4. Defining different Object Properties



The screenshot shows the Protégé ontology editor interface. The left pane displays the ontology hierarchy, and the right pane shows the definition of the `isEmployedBy` property.

Property Definition:

- Property:** `isEmployedBy`
- Domain:** `Inquirer`
- Range:** `Hospital`
- Disjoint With:** (None)
- SuperProperty Of (Chain):** (None)

Instances:

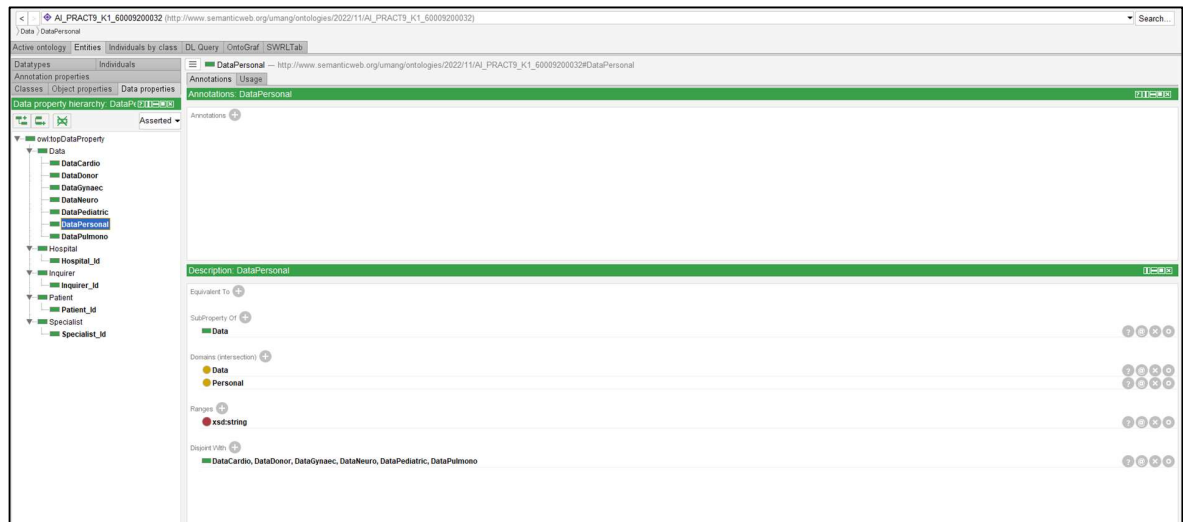
- `Dr_Bulma isEmployedBy Hospital_Forts`
- `Dr_Lodaya isEmployedBy Hospital_Forts`
- `Dr_Senju isEmployedBy Hospital_Forts`

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



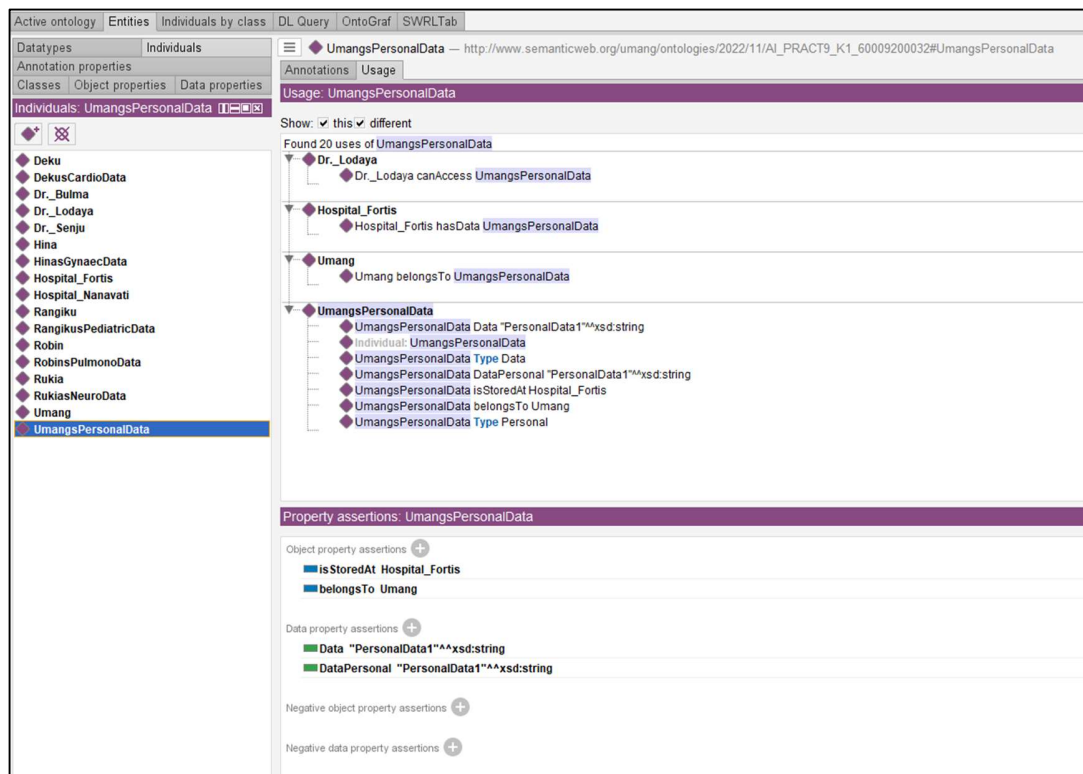
Department of Computer Science and Engineering (Data Science)

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)

7. Summary – Individuals by Classes

a. Individuals by Class Data

The screenshot shows the Protege interface with the 'Data' class selected. The left pane displays the class hierarchy under 'Data', including 'Data', 'Hospital', 'Inquirer', and 'Patient'. The right pane shows the 'Usage' tab for 'DekusCardioData', listing 20 uses. The bottom pane shows the 'Direct instances' of 'DekusCardioData', including 'HinasGynaecData', 'RangikusPediatricData', 'RobinsPulmonologyData', 'RukiasNeuroData', and 'UmangsPersonalData'. The 'Description' and 'Property assertions' panes are also visible.

b. Individuals by Class Hospital

The screenshot shows the Protege interface with the 'Hospital' class selected. The left pane displays the class hierarchy under 'Hospital', including 'Data', 'Hospital', 'Inquirer', and 'Patient'. The right pane shows the 'Usage' tab for 'Hospital_Fortis', listing 42 uses. The bottom pane shows the 'Direct instances' of 'Hospital_Fortis', including 'Hospital_Fortis' and 'Hospital_Nanavati'. The 'Description' and 'Property assertions' panes are also visible.



Department of Computer Science and Engineering (Data Science)

c. Individuals by Class Inquirer

The screenshot shows the OWL Inquirer tool interface. The top navigation bar includes 'Active ontology', 'Entities', 'Individuals by class', 'DL Query', 'OntoGraf', and 'SWRLTab'. The 'Class hierarchy: Inquirer' is displayed on the left, showing a tree structure with 'Data', 'Hospital', 'Inquirer', and 'Patient'. The 'Direct instances: Dr._Lodaya' are listed as 'Dr._Bauma', 'Dr._Lodaya', and 'Dr._Senju'. The 'Usage: Dr._Lodaya' section on the right shows 24 uses of the class, including 'Dr._Lodaya Type Gynaecologist', 'Dr._Lodaya Specialist_Id "Personal1"^^xsd:string', 'Dr._Lodaya Type Inquirer', 'Dr._Lodaya Specialist_Id "Gynaec1"^^xsd:string', 'Dr._Lodaya isEmployedBy Hospital_Fortis', 'Individual: Dr._Lodaya', 'Dr._Lodaya Specialist "Personal1"^^xsd:string', 'Dr._Lodaya canAccessGynaec HinasGynaecData', 'Dr._Lodaya Type Specialist', 'Dr._Lodaya Specialist "Gynaec1"^^xsd:string', and 'Dr._Lodaya canAccess UmangsPersonalData'. The 'Description: Dr._Lodaya' section shows the types 'Gynaecologist', 'Inquirer', and 'Specialist'. The 'Property assertions: Dr._Lodaya' section shows object property assertions 'isEmployedBy Hospital_Fortis', 'canAccessGynaec HinasGynaecData', and 'canAccess UmangsPersonalData', and data property assertions 'Specialist_Id "Personal1"^^xsd:string', 'Specialist_Id "Gynaec1"^^xsd:string', 'Specialist "Personal1"^^xsd:string', and 'Specialist "Gynaec1"^^xsd:string'.

d. Individuals by Class Patient

The screenshot shows the OWL Inquirer tool interface. The top navigation bar includes 'Active ontology', 'Entities', 'Individuals by class', 'DL Query', 'OntoGraf', and 'SWRLTab'. The 'Class hierarchy: Patient' is displayed on the left, showing a tree structure with 'Data', 'Hospital', 'Inquirer', and 'Patient'. The 'Direct instances: Umang' are listed as 'Deku', 'Hina', 'Rangiku', 'Robin', 'Rukia', and 'Umang'. The 'Usage: Umang' section on the right shows 14 uses of the class, including 'Umang Type Patient', 'Umang isAdmittedTo Hospital_Fortis', 'Umang Patient_Id "PersonalPatient1"^^xsd:string', 'Umang Patient "PersonalPatient1"^^xsd:string', 'Umang belongsTo UmangsPersonalData', and 'Individual: Umang'. The 'Description: Umang' section shows the type 'Patient'. The 'Property assertions: Umang' section shows object property assertions 'isAdmittedTo Hospital_Fortis' and 'belongsTo UmangsPersonalData', and data property assertions 'Patient_Id "PersonalPatient1"^^xsd:string' and 'Patient "PersonalPatient1"^^xsd:string'.



Department of Computer Science and Engineering (Data Science)

8. Running Query

AI_PRACT9_K1_60009200032 (http://www.semanticweb.org/umang/ontologies/2022/11/AI_PRACT9_K1_60009200032)

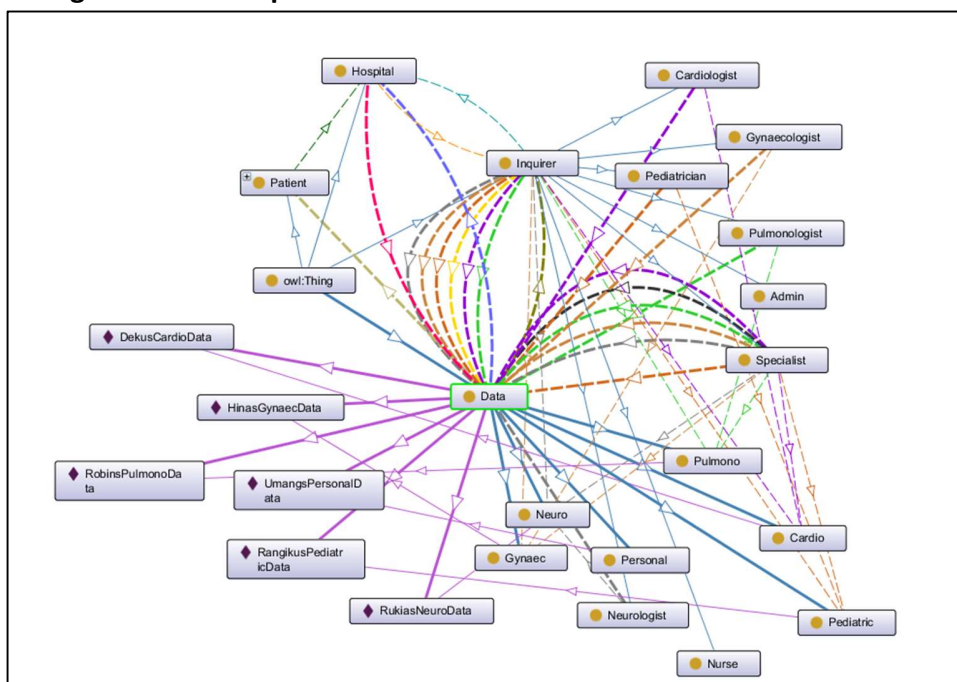
Active ontology: Entities | Individuals by class | DL Query | OntoGraf | SWRLTab

Name	Rule
General	Specialist(?a) * Regular(?y) * belongsTo(?a, ?y) * isEmployedBy(?x, ?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.