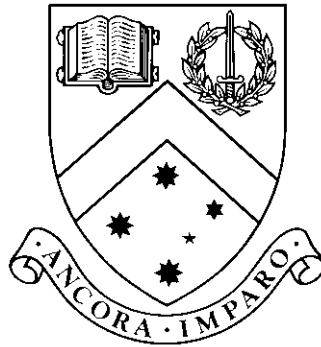


# Effective Visualisations of Large Ontologies and Associations

by

**Ying Yang**



**Thesis**

Submitted by Ying Yang

for fulfillment of the Requirements for the Degree of  
**Doctor of Philosophy (0190)**

Supervisor: Dr Michael Wybrow

Associate Supervisor: Dr Yuan-Fang Li

Associate Supervisor: Dr Tobias Czauderna

**Faculty of Information Technology**  
**Monash University**

February, 2021

© Copyright

by

Ying Yang

2021



# Copyright Notice

© Ying Yang (2021)

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

# Effective Visualisations of Large Ontologies and Associations

## Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

---

Ying Yang  
February 7, 2021

*For Mum and Dad.*

# Acknowledgments

I will forever be grateful to my supervisors Dr. Michael Wybrow, Dr. Yuan-Fang Li and Dr. Tobias Czauderna for their constant guidance and support. They have been a great resource of encouragement, critique and inspiration. I feel extraordinarily lucky to have their dedication, patience and enthusiasm during this challenging adventure. I would also like to thank them for their thoughtful advice on the amendments for this thesis.

Many thanks go to my panel members, Professor Tim Dwyer, Associate Professor Bernhard Jenny and Dr. Sarah Goodwin for their valuable time and insightful suggestions. They have greatly helped me to shape my thinking on my research. I also thank my thesis examiners Professor Michel Dumontier and Professor Niklas Elmqvist for their constructive feedback on this thesis.

Thanks to all the members of Data Visualisation and Immersive Analytics Lab at Monash University. I deeply appreciate the friendly nature of the lab. My thanks also go to my wonderful fellows in the lab, for providing a lively environment and generously sharing their research.

I would like to thank my collaborators especially Associate Professor Yongqun He for his engagement of discussions on the various use cases. The members of He Group in University of Michigan and Cognitive Intelligence Lab in Southeast University are also gratefully acknowledged.

Thanks to Monash University and Faculty of Information Technology for granting the scholarships, without which it would not have been possible to pursue this research.

I would like to extend my appreciation to the administration team in the faculty, especially Ms Danette Deriane, Ms May Chew and Ms Kerry McManus for their unbounded support and assistance.

It is a pleasure to thank my office mates especially Ehsan Shareghi Nojehdeh, Kai Siong Yow, Xuhui Zhang, Dinithi Sumanaweera and Shuang Yu for providing an enjoyable working environment. I would also like to thank my dear friends Wudhichart Sawangphol, Ye Zhu, Sameen Maruf, Ming Liu, Zihao Wen, Chang Wei Tan and Ke Yang, who have generously shared their wisdom and experience.

My deepest gratitude goes to my parents, for their unconditional love and support throughout my study and life.

Ying Yang

*Monash University  
February 2021*

# Contents

<b>Copyright Notice</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>vi</b>
<b>List of Tables</b>	<b>xi</b>
<b>List of Figures</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation	2
1.2 Research Objectives	5
1.3 Research Methodology	6
1.4 Contributions	6
1.5 Thesis Structure	8
<b>2 User Needs</b>	<b>9</b>
2.1 Ontology and Association	9
2.1.1 Biomedical Ontologies	10
2.1.2 Multiple Ontologies	11
2.2 Exploratory Interviews	12
2.3 Large Ontologies	13
2.3.1 Use Cases	13
2.3.2 Visualisation Requirements	15
2.3.3 Current Approach	16
2.4 Homogeneous Associations	16
2.4.1 Use Cases	16
2.4.2 Visualisation Requirements	17
2.4.3 Current Approach	18
2.5 Heterogenous Associations	20
2.5.1 Use Cases	20
2.5.2 Visualisation Requirements	21
2.5.3 Current Approach	23
2.6 Summary	25
2.7 Conclusion	28
<b>3 Visualisation Techniques</b>	<b>29</b>
3.1 Hierarchy Visualisation	29
3.2 Ontology Visualisation	33
3.2.1 Connection	34
3.2.2 Containment	36
3.2.3 Adjacency	38
3.2.4 Multiple Views	38

3.2.5	Summary . . . . .	44
3.3	Visual Compression . . . . .	46
3.3.1	Aggregation . . . . .	46
3.3.2	Clustering . . . . .	46
3.3.3	Simplification . . . . .	47
3.4	Conclusion . . . . .	50
<b>4</b>	<b>Visualising Large Ontologies . . . . .</b>	<b>51</b>
4.1	Design Space Exploration . . . . .	51
4.1.1	Nested Treemap . . . . .	51
4.1.2	Sliced Treemap . . . . .	52
4.1.3	Icicle Plot Variant . . . . .	53
4.1.3.1	Hybrid Icicle Plot and Treemap . . . . .	53
4.1.3.2	GridPlot . . . . .	54
4.2	OntoPlot . . . . .	55
4.2.1	Visual Representation . . . . .	55
4.2.2	Visual Compression . . . . .	56
4.2.2.1	Glyphs for Visual Summary . . . . .	56
4.2.2.2	Design Space of Glyphs . . . . .	58
4.2.3	Interaction . . . . .	58
4.2.3.1	Collapse and Expansion . . . . .	58
4.2.3.2	Pinning Class Label . . . . .	59
4.2.3.3	Search . . . . .	60
4.3	Conclusion . . . . .	60
<b>5</b>	<b>Visualising Homogeneous Associations . . . . .</b>	<b>63</b>
5.1	Prototype . . . . .	63
5.1.1	Visual Interface . . . . .	63
5.1.1.1	Colour Key . . . . .	64
5.1.2	Interaction . . . . .	64
5.1.2.1	Scrolling . . . . .	65
5.1.2.2	Information Pop-ups for Classes . . . . .	65
5.1.2.3	Class Selection . . . . .	65
5.2	Prototype Evaluation . . . . .	66
5.2.1	Study Design . . . . .	66
5.2.2	Comparison with WebVOWL . . . . .	67
5.2.3	Tasks . . . . .	68
5.2.4	Hypotheses . . . . .	69
5.2.5	Datasets . . . . .	71
5.2.6	Procedure . . . . .	71
5.2.7	Participants and Apparatus . . . . .	72
5.2.8	Results . . . . .	72
5.2.9	Discussion . . . . .	77
5.3	Prototype Refinement . . . . .	78
5.3.1	Visual Representation . . . . .	78
5.3.1.1	Different Separation of Boxes . . . . .	78
5.3.1.2	Compression of Leaf Nodes . . . . .	79
5.3.1.3	Class Labels . . . . .	79
5.3.1.4	Association Labels . . . . .	80
5.3.2	Interaction . . . . .	81
5.3.2.1	Expanding and Collapsing . . . . .	81
5.3.2.2	Class Selection and Focus Mode . . . . .	81

5.3.2.3	Search . . . . .	83
5.4	Expert User Study . . . . .	83
5.4.1	Study Structure . . . . .	83
5.4.2	Participants and Apparatus . . . . .	84
5.4.3	Results . . . . .	84
5.4.4	Discussion . . . . .	87
5.5	Visualisation Extensions . . . . .	88
5.5.1	Hill Glyph . . . . .	88
5.5.2	Hidden Structure Visual Summary . . . . .	89
5.5.3	Scrollable Class Labels . . . . .	89
5.5.4	Non-overlapping Association Labels . . . . .	90
5.5.5	Minimap . . . . .	91
5.5.6	Import and Export . . . . .	91
5.6	Performance Measure . . . . .	92
5.7	Conclusion . . . . .	92
<b>6</b>	<b>Visualising Heterogeneous Associations . . . . .</b>	<b>95</b>
6.1	Visual Design . . . . .	95
6.1.1	Visualising Multiple Properties . . . . .	95
6.1.1.1	Pie Glyphs Attempt . . . . .	96
6.1.1.2	Association Labels . . . . .	97
6.1.1.3	Union and Intersection . . . . .	98
6.1.2	Multiple Class Selection . . . . .	99
6.1.2.1	Highlighting Property List . . . . .	99
6.1.2.2	Colour Glow . . . . .	100
6.1.2.3	Grid Labels . . . . .	101
6.1.2.4	Show Common Node . . . . .	103
6.1.2.5	Multi-Focus Mode . . . . .	103
6.1.2.6	Search Function Refinement . . . . .	103
6.2	Case Studies . . . . .	105
6.2.1	Comparing properties to explore associations between microbiomes and diseases . . . . .	105
6.2.1.1	Comparing microbe interactions in different diseased pa- tient guts . . . . .	105
6.2.1.2	Comparing microbe interactions in different diseased species	108
6.2.1.3	Comparing microbe interactions in different diseased hu- man hosts . . . . .	112
6.2.2	Linking properties to explore associations between drugs and their adverse events . . . . .	117
6.2.2.1	Analysing agonist and antagonist neuropathy-inducing drugs and their targets in ODNAE . . . . .	118
6.2.2.2	Analysing cardiovascular drugs and their adverse events based on drug mechanism of action classification in OCV- DAE . . . . .	121
6.3	Summary . . . . .	125
6.4	Discussion . . . . .	125
6.5	Conclusion . . . . .	127
<b>7</b>	<b>Conclusions . . . . .</b>	<b>129</b>
7.1	Contributions . . . . .	129
7.2	Limitations . . . . .	130
7.2.1	Evaluation . . . . .	130

7.2.2	Scalability . . . . .	130
7.3	Future Directions . . . . .	131
7.3.1	Ontology Class Lables . . . . .	131
7.3.2	Ontology Property Composition . . . . .	131
7.3.3	Many-to-Many Ontology Association . . . . .	132
7.3.4	Ontology Axioms . . . . .	132
7.3.5	Ontology Instances . . . . .	132
7.3.6	Multiple Ontologies . . . . .	132
7.4	Closing Remarks . . . . .	133
<b>Vita . . . . .</b>		<b>135</b>
<b>References . . . . .</b>		<b>135</b>



# List of Tables

2.1	Common use cases when working with ontologies. . . . .	14
2.2	Visualisation design requirements for ontologies in the context of identified use cases. . . . .	15
2.3	Common use cases when working with homogeneous associations in ontologies.	17
2.4	Visualisation design requirements for homogeneous associations in the context of identified use cases. . . . .	18
2.5	Common use cases when working with heterogeneous associations of ontologies. . . . .	22
2.6	Visualisation design requirements for heterogeneous associations in the context of identified use cases. . . . .	23
2.7	Summary of the interview data analysis outcome. . . . .	26
3.1	Examples of hierarchical structure representations . . . . .	30
3.2	Summary of ontology visualisation tools . . . . .	45
5.1	Tasks in the experiment. . . . .	70
5.2	Summary of statistical significance (Wilcoxon test $p$ values) of accuracy difference as summarised in Figure 5.8, for both ontologies and each of the two individual ontologies. . . . .	73
5.3	Summary of statistical significance (Whitney-Mann test $p$ values) of differences in completion time as summarised in Figure 5.9, for both ontologies and each of the two individual ontologies. . . . .	74
5.4	Summary of statistical significance (Wilcoxon test) of differences in difficulty and confidence ratings, summarised in Figures 5.10a and 5.10b. . . . .	75
5.5	Overall summary of statistical significance of results. The tool mentioned in the columns outperforms the other in terms of the given metric (O: OntoPlot, P: Protégé). . . . .	77
5.6	Summary of average difficulty and confidence ratings as shown in Figures 5.22a and 5.22b. . . . .	85
5.7	Summary of statistical significance for the expert user study results. The tool mentioned in the columns outperforms the other in terms of the given metric (O: OntoPlot, P: Protégé). . . . .	87
5.8	Summary of OntoPlot performance time. . . . .	93

# List of Figures

1.1	Simple illustration of the ontology relations with the Pizza ontology (Drummond et al., 2007). The subclass relations are drawn with solid lines while the associations are drawn with dashed lines. . . . .	1
1.2	Simple illustrations of basic hierarchical representations, visualising the same hierarchy. . . . .	3
1.3	The user-centred research methodology for this research adapting the design process model (Takeda et al., 1990). . . . .	7
2.1	A simple illustration hierarchy. . . . .	14
2.2	The interface of Protégé, visualising the Pizza ontology. . . . .	16
2.3	An example of how domain experts manually annotate screenshots of the Protégé indented list to show the class effect: the <b>Fatty Acids</b> [Chemical/Ingredient] class (left) has association with all the leaf classes in the adverse event subtree (right) (Wang et al., 2017). . . . .	19
2.4	An example of the SPARQL query written by domain experts (Guo et al., 2016). . . . .	19
2.5	An example of the manual approach to show association strength within an ontology: association numbers written next to branches of the hierarchy on a screenshot of Protégé indented list view (Guo et al., 2016). . . . .	20
2.6	A bubble chart alongside an annotated Protégé indented list was used by domain experts to show the association strength for the ontology classes (Liu et al., 2017). . . . .	20
2.7	An illustration ontology adapting the Pizza ontology to illustrate simple heterogeneous association use cases. . . . .	22
2.8	An example of how domain experts present three heterogeneous associations in the ontology hierarchy by annotating screenshots of the Protégé indented list view (Wang et al., 2017). @: the indented list view of ontology hierarchy, ⑤: the class description view listing associations for the selected class <b>FLUVASTATIN NA 20MG CAP</b> [VA Product]. . . . .	24
2.9	The heatmap used by experts to show the association strength between adverse events and molecular entities linked by drug products (Guo et al., 2016). . . . .	24
2.10	Examples of using screenshots of Protégé indented list to show heterogeneous associations on ontology hierarchies. . . . .	25
3.1	Nested treemap and cascaded treemap (Lü and Fogarty, 2008). . . . .	32
3.2	An example of the one dimensional treemap representation in ArcTrees (Neumann et al., 2005). . . . .	32
3.3	Onto Design Graphics (ODG) (Silva-López et al., 2014) based on Unified Modeling Language (UML) component. . . . .	35

3.4	Visual Web Ontology Language (VOWL) (Lohmann, Negru, Haag and Ertl, 2014) based on a well-defined user-friendly set of graphical primitives and colour scheme. . . . .	35
3.5	KC-Viz visualisation (Motta et al., 2011) . . . . .	36
3.6	OWLviz visualisation (Noy et al., 2000) . . . . .	36
3.7	NeXO Web visualisation of Gene Ontology (Dutkowski et al., 2013) . . . .	37
3.8	Treemap of GO (Babaria, 2004). Size shows the amount of fold change for genes and colour shows gene acidity: green (basic) or red (acidic). . . . .	37
3.9	CropCircles visualisation (Wang and Parsia, 2006) . . . . .	38
3.10	Indented list in Protégé, showing the Pizza ontology. . . . .	39
3.11	ERSF visualisation (Jia et al., 2010). . . . .	39
3.12	Jambalaya interface in Protégé (Storey et al., 2001) . . . . .	40
3.13	Knoocks visualising an ontology hierarchy and associations (Jurcik, 2012). .	41
3.14	AlViz interface (Lanzenberger et al., 2010) . . . . .	41
3.15	OntoViewer interface (da Silva et al., 2012) . . . . .	42
3.16	The three-view visualisation proposed by Kuhar and Podgorelec (2012), visualising an ontology having 45 classes with a class selected. . . . .	43
3.17	Ontology visualisation in Dunnart (Jiao et al., 2013) . . . . .	43
3.18	OntoTrix interface (Bach et al., 2011). . . . .	44
3.19	An example of Lineage aggregation (Nobre et al., 2018). The interesting nodes are coloured black. . . . .	47
3.20	Clustering of a network (Archambault et al., 2010). . . . .	47
3.21	Glyph variations (Fuchs et al., 2014): (a) contour only, (b) structure only, (c) contour + structure. . . . .	48
3.22	Fan, connector and clique motif (top) and their simplification glyphs (bottom) (Dunne and Shneiderman, 2013). . . . .	48
3.23	Motif simplification (Dunne and Shneiderman, 2013): (a) the original network (b) a simplified version using fan and connector glyphs. . . . .	49
3.24	SpaceTree showing a hierarchy (Plaisant et al., 2002). . . . .	49
3.25	DOI Tree visualising a hierarchy with 600,000 nodes (Heer and Card, 2004). 50	
4.1	Nested treemap for the OAE ontology. . . . .	52
4.2	Three levels of a sliced treemap displaying the OAE ontology. . . . .	53
4.3	Comparison of an icicle plot and the hybrid icicle plot and treemap. . . . .	54
4.4	Comparison of the hybrid icicle plot and treemap and GridPlot. . . . .	54
4.5	Comparison of GridPlot, OntoPlot and icicle plot. . . . .	56
4.6	Examples of visual compression. . . . .	57
4.7	Sketch of different settings of glyphs for compressed hierarchy subtrees. . .	59
4.8	An example of pinning a series of class labels from the ancestors to a leaf class of interest. . . . .	60
4.9	When the user enters the search term “process” and selects the class <b>cardiac arrhythmia (process)</b> from the matching classes list, the label of the <b>cardiac arrhythmia (process)</b> class is displayed. As all the classes in the <b>cardiac arrhythmia (process)</b> subtree contain the term “process”, their glyphs are surrounded by additional circles. Note, the search panel consisting of the search field and the matching classes list is cropped and brought together in this image. . . . .	60

5.1	The visual interface of the OntoPlot prototype. ① is the property list. ② is an example of a subtree compressed as a triangle. ③ is a class that has associations. Classes with associations are highlighted in colour based on the key ④ on the right-hand side. Note, the search field ⑤ introduced in Section 4.2.3.3 is shown on the top-right. . . . .	64
5.2	Colours for associations. Unique colours show the minimum and maximum number of associations. . . . .	64
5.3	A pop-up window shows association information while hovering over classes.	65
5.4	The visualisation when the class FMA_3710 is selected. . . . .	66
5.5	Protégé interface as configured for use in the study. ①: Class pane, ②: Object Property pane, ③: Class Description view, ④: Class Usage view, ⑤: Property Usage view. . . . .	67
5.6	Visual comparison between WebVOWL and OntoPlot on the Pizza ontology. Note, as the Pizza ontology does not contain multiple inheritance, there are no duplicated classes in OntoPlot. . . . .	68
5.7	When visualising the ODNAE ontology with 1,545 classes and 810 associations in WebVOWL. . . . .	69
5.8	Mean accuracy for each tool per task. . . . .	73
5.9	Mean completion time for each tool per task. . . . .	74
5.10	Participants' rating of the two tools: (a) difficulty rating for each group of tasks, (b) confidence rating for each group of tasks. . . . .	75
5.11	Participants' rating of the two tools: (a) preference rating for each group of tasks, and (b) learning effort rating. . . . .	76
5.12	Overview of the refined visualisation. . . . .	78
5.13	Solid lines are shown between different subtree boxes, e.g. ①. A partial and faint line is shown between sibling boxes, e.g. ②. . . . .	79
5.14	An example of refined compression of leaf nodes. . . . .	79
5.15	Parent classes are labelled where possible. . . . .	80
5.16	Examples of association labels. . . . .	80
5.17	An example of highlighting portions when double-clicking a triangle glyph to expand a subtree. . . . .	81
5.18	The collapsed subtree ① containing association classes is highlighted with a coloured glow. The collapsed subtree ② containing the selected class and association classes is highlighted with a pulsing red circle and a coloured glow. The red glow of ② and yellow glow of ① indicate the maximum number of associations inside the subtrees. . . . .	82
5.19	When a class is selected, OntoPlot highlights it ① and shows additional information and controls ②. . . . .	82
5.20	The <i>focus mode</i> for class AE severity G2 provides a compressed view that allows users to focus on an interesting subset of the ontology based on the associations for just this class. Note the notification bar at the top of the view. . . . .	83
5.21	Participants' performance using the two tools in the expert user study. . . . .	85
5.22	Participants' rating of the two tools in the expert user study. . . . .	85
5.23	Participants' rating of the two tools in the expert user study. . . . .	86
5.24	An example of hill glyph visual compression. . . . .	89
5.25	Visual summary of the hidden structure in a triangle glyph. . . . .	89
5.26	A situation when ancestor class circles are scrolled out of the view. ① the scrolled labels are automatically repositioned during scrolling to stay visible on screen. . . . .	90

5.27	An example of automatic shifting of overlapping association labels in (a) to get non-overlapping association labels in (b). . . . .	90
5.28	The minimap of the main visualisation is displayed at the bottom. . . . .	91
6.1	The visualisation with two properties selected. ① is the cropped property list showing property <i>BFO_0000066</i> and <i>BFO_0000082</i> are selected. . . . .	96
6.2	The visualisation with three properties selected. Note, as there is no association linked by the third property <i>BFO_0000167</i> to the class shown in the callout, sector 3 of that class is coloured white. . . . .	96
6.3	The association labels when three properties are selected. Box 1 is responsible for the first property <i>BFO_0000066</i> , and so on. . . . .	97
6.4	An example of the visualisation when a user mouse hover over a box of an association label. . . . .	98
6.5	When a user double-clicks the second box ② on the association label of the <i>FMA_7088</i> class, only the classes having <i>BFO_0000082</i> association with <i>FMA_7088</i> are labelled (for example, ③). . . . .	98
6.6	<i>Union</i> and <i>intersection</i> for multiple properties. . . . .	99
6.7	When the classes marked as ① and ② are selected, the property list is updated. Both of the <i>BFO_0000054</i> and <i>BFO_0000055</i> properties are moved to the top of the list, with <i>BFO_0000054</i> having a dark grey background, indicating <i>BFO_0000054</i> is applied to both selected classes. When a user hovers over the <i>BFO_0000055</i> property on the property list, the class ③ is enlarged, indicating <i>BFO_0000055</i> is only applied to class ③. . . . .	100
6.8	When selecting the class marked as ①, the classes that have associations with it are labelled and coloured (for example, ④). The classes that have associations with other classes rather than ① are surrounded by a coloured glow (for example, ⑤). . . . .	100
6.9	Grid labels of the selected classes <i>DOID_1869</i> and <i>DOID_8805</i> , with three properties <i>BFO_0000054</i> , <i>BFO_0000055</i> and <i>BFO_0000172</i> selected. . . . .	101
6.10	Five classes are selected in the visualisation and five grid labels are generated. The numbers ①②③④⑤ labelling the grid labels match the numbers labelling the selected classes, indicating the corresponding grid label and class respectively. Note, leader lines are shown for each grid label, though very faintly. . . . .	102
6.11	Highlighting is displayed when users hover the mouse over a cell on a grid label. Note, the leader line of that grid label is darkened in the visualisation. Others are faintly visible. . . . .	103
6.12	The <i>show common node</i> mode identifies the common node ⑥ for the selected classes ③ and ④, indicating ⑥ is linked by one of the selected properties <i>BFO_0000055</i> (see arrow) to both ③ and ④. . . . .	104
6.13	The <i>multi-focus mode</i> for the selected classes ③ and ④. In <i>focus mode</i> the visualisation is compressed and only shows the association classes ⑦ and ⑧ for ③ and ④ together with their ancestor classes. . . . .	104
6.14	Summary of key features of OntoPlot for heterogeneous association use cases.	126
7.1	Illustration of possible further compression for sibling leaf association classes (highlighted in the blue boxes) and <i>uninteresting</i> intermediate level classes (highlighted in the red boxes). . . . .	131

# Effective Visualisations of Large Ontologies and Associations

Ying Yang  
ying.yang@monash.edu  
Monash University

Supervisor: Dr. Michael Wybrow  
michael.wybrow@monash.edu  
Associate Supervisor: Dr. Yuan-Fang Li  
yuanfang.li@monash.edu  
Associate Supervisor: Dr. Tobias Czauderna  
tobias.czauderna@monash.edu

## Abstract

Ontologies are widely used in many domains, such as biomedicine, agronomy, e-government and bibliometrics, where they serve as shared vocabularies. They mean to capture comprehensive domain knowledge. Ontologies are explicit and formal representations, describing concepts and the relationships between these concepts. The large ones can contain hundreds of thousands or even millions of concepts and relationships. Their structure can be complex, constructed by subsumption hierarchical relationships and non-hierarchical *associations*. Their hierarchy structure features multiple inheritance, demanding thoughtful choice of representation for the ontology hierarchy. The non-hierarchical associations define a variety of rich information in addition to the subsumption hierarchy, making the comprehension of ontology relationships more difficult.

Visualisations are useful to understand such data. The large size and complex structure of ontologies and associations require effective visualisations for exploring and understanding their underlying data, and also pose challenges for the design and development of such visualisations. Effective visualisations should satisfy user needs and support user activities. The non-hierarchical associations lay the foundation for performing ontology analysis. While there is demand to emphasise these associations, existing ontology visualisation tools generally treat the hierarchical structure as first-class citizens, giving much less attention to the associations.

This research explores the approaches for visualising large ontology hierarchies and associations. It proposes a visualisation, OntoPlot, specifically designed to facilitate the exploration of ontology associations, whilst still preserving an ontology's underlying large hierarchical structure.

OntoPlot constructs a space-efficient icicle plot like hierarchy representation and employs visual compression techniques to accommodate large ontology hierarchies. It effectively depicts complex associations on the ontology hierarchy in a way that the associations

can stand out from the hierarchy, while the overview and details of associations are easily available. It offers diverse interactivity, incorporating many desired user activities.

The usability of OntoPlot is explored via several user studies, comparing OntoPlot with the de facto ontology editor Protégé, one of which is with domain experts. The results confirm that OntoPlot attains the design goals for association-related tasks and is strongly favoured by domain experts and other users. OntoPlot is also evaluated with a case study in the context of biomedicine, demonstrating its effectiveness for assisting domain experts to accomplish complex association tasks.





# Chapter 1

## Introduction

*Ontology*, originating from philosophy, studies the essence of things that exist in nature, in particular the identification and categorisation of these things and their relations (Gruber et al., 1993). It has subsequently been given a definition in computer science as a formal technical specification of conceptualisation that can be understood by computer (Gruber et al., 1993).

In computer science, an ontology describes concepts and the relationships between these concepts. A concept is a class<sup>1</sup> of individuals. A relationship is denoted as a binary relation.

The backbone relations in an ontology are subclass relations, which define the subsumption hierarchical structure for the ontology. For example, *Pizza* *subClassOf* *Food* presents a subclass relation (see Figure 1.1). Besides the subclass relations, the binary relations can be linked by properties, expressing relevance (Staab and Studer, 2009). For example, the classes *MushroomPizza* and *MushroomTopping* are linked by the property *hasTopping* to express the relevant relation that places a restriction (see Figure 1.1). In this research, the term *association* is used to define these restrictions that involve a property.

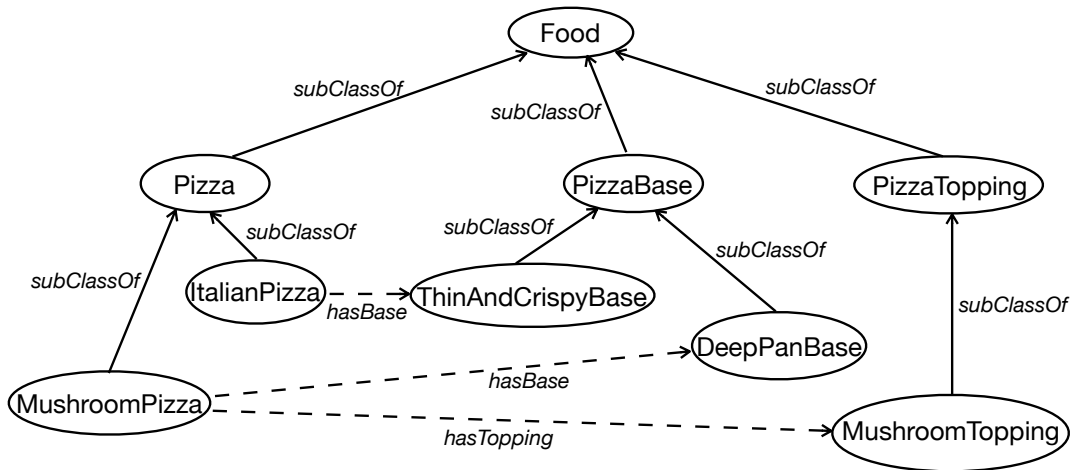


Figure 1.1: Simple illustration of the ontology relations with the Pizza ontology (Drummond et al., 2007). The subclass relations are drawn with solid lines while the associations are drawn with dashed lines.

---

<sup>1</sup>In the rest of this thesis the term class is used instead of concept.

Given a collection of associations, the terms *homogeneous* and *heterogeneous* are used throughout this thesis to denote the uniformity and non-uniformity of associations in the collection. If the collection of associations contains only the same property, they are called *homogeneous* associations. For example, in Figure 1.1 the association that links *MushroomPizza* to *DeepPanBase* with *hasBase* and the one that links *ItalianPizza* to *ThinAndCrispyBase* with *hasBase* are homogeneous associations, connected by the same property *hasBase*. When the collection involves different properties, they are *heterogeneous* associations. In Figure 1.1 the association consisting of *MushroomPizza* (class), *hasBase* (property) and *DeepPanBase* (class) and the one consisting of *MushroomPizza* (class), *hasTopping* (property) and *MushroomTopping* (class) are heterogeneous associations. The homogeneous and heterogeneous associations in an ontology capture a variety of rich information in addition to the subsumption hierarchy.

Ontologies are widely used in diverse areas, such as biomedicine, agronomy, e-government and bibliometrics (d’Aquin and Noy, 2012). Many large ontologies have been developed in last two decades, which contain thousands or even tens or hundreds of thousands classes. For example in the biomedical area, BioPortal (Noy et al., 2009; Salvadores et al., 2013), a comprehensive biomedical ontology repository, currently contains 763 ontologies with a total of almost 10 million classes.<sup>2</sup> These include the influential Gene Ontology (Ashburner et al., 2000), which has close to 50,000 classes, and the SNOMED CT ontology (Stearns et al., 2001), which currently has more than 340,000 classes.

Ontologies also have complex structures. Their hierarchical structure usually contains *multiple inheritance*<sup>3</sup>, demanding efficient means to reveal all direct ancestors of a class to users. The homogeneous and heterogeneous ontology associations, introducing additional relationships and structure complexity, make the comprehension and use of ontologies and their associations more difficult.

Visual tool support for the effective interrogation of such large and complex data is essential to both ontology users and ontology developers. Visualisations provide visual representations of data, often designed to convey knowledge and reduce user perceptual and cognitive effort. An effective visualisation can help users get insights into data, explore datasets and perform tasks (Munzner, 2014).

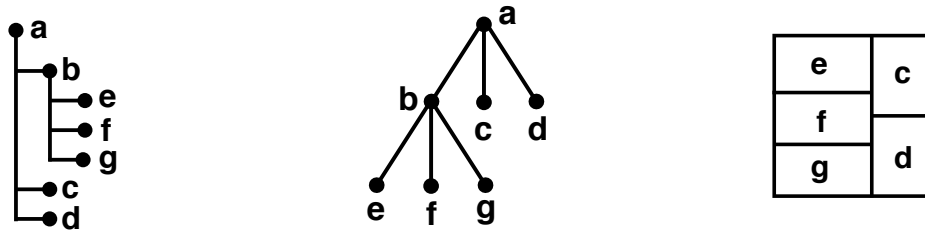
## 1.1 Motivation

As a response to this important need, many ontology visualisation systems have been developed in recent decades, as explained by several survey papers (Katifori et al., 2007; Saghafi, 2016; Dudáš et al., 2018). Given the central importance of subsumption relationships in defining an ontology, most of the existing visualisation systems rightly treat hierarchies as first-class citizens. For example, Protégé (Noy et al., 2000), the de facto ontology editor, shows the subsumption hierarchy of the ontology in an indented tree (see Figure 1.2a).

<sup>2</sup><https://bioportal.bioontology.org/>

<sup>3</sup>Where classes have multiple superclasses.

However, the basic hierarchical representations are not adequate for showing large and complex ontologies effectively. While the number of classes increases, an indented tree will become very tall, requiring considerable time for navigation. Other common hierarchy representations, like node-link layered trees (Reingold and Tilford, 1981), are organised and intuitive, but do not scale well (see Figure 1.2b). Basic treemaps (Shneiderman, 1992) perform well in scalability, but are not good at showing hierarchical structures (see Figure 1.2c). Also, for ontologies containing multiple inheritance, which is normally solved by duplicating a term under each of its parents or using multiple edges to link a term to all its parents, the visual redundancy or interleaving of edges is inevitable. Thus, there is a need for defining better visualisations for such data.



(a) Indented tree, represents classes as nodes and subclass relations as edges with relative vertical positions. The nodes are indented horizontally to the right corresponding to their depth in the tree.

(b) Node-link layered tree, represents classes as nodes and subclass relations as edges. Child nodes are placed below their parent nodes.

(c) Treemap, represents classes as rectangles and places all the child classes enclosed within the area assigned to their parent classes.

Figure 1.2: Simple illustrations of basic hierarchical representations, visualising the same hierarchy.

This leads to the motivation for this research: finding effective approaches for visualising complex large ontologies and associations. Since different hierarchy representations trade off structure for scalability, it is natural to ask whether one can combine their strength to develop a more space-efficient representation to accommodate large ontologies.

Visual compression techniques can provide a solution to reduce visual occlusion and visualisation complexity. They are often used with interaction techniques, which allow users to investigate and explore the details of information hidden in compressed elements. In recent years, the utilisation of glyphs for representations of groups of nodes in graphs and hierarchies has drawn some attention since it allows for a more compact representation by compressing or simplifying the topology (Shneiderman and Dunne, 2012; Plaisant et al., 2002). However, arguments have been made that these abstract glyphs hide the details of structure and the expansions of nodes require users' explicit actions, which slows down the process of assimilating unfamiliar datasets. It is possible to argue that with more expressive glyphs or other visual summary techniques, the compressed structure can be understood more easily. Also, automatic mechanisms like Degree-of-Interest (DOI) trees (Card and Nation, 2002), which calculate data degree-of-interest values and employ the focus+context method (Furnas, 1986) to choose data portions displayed on

constrained screen, can be explored to define compressed and expanded elements, so as to reduce effort by users.

When focusing on use of associations rather than tree layouts, one might naturally suggest the use of network layouts that give freedom to arrange the ontology classes based on their connectivity. One typical example is WebVOWL (Lohmann, Link, Marbach and Negru, 2014), which models all the relations in an ontology using the node-link representation with force-directed layout (Fruchterman and Reingold, 1991). It can be argued that this approach does not scale up to large ontologies, resulting the infamous “hair-ball” effect (Jankun-Kelly et al., 2014). An earlier tool, Jambalaya (Storey et al., 2001), visually differentiates hierarchical relationships and non-hierarchical associations, placing the ontology classes on the tree layouts and drawing links between classes on top of the hierarchy to represent associations, with different colours denoting different association types. It was noted that in Jambalaya the interweaving of links dramatically reduces the readability of visualisation. Thus, an effective approach needs to be defined to properly organise the heterogeneous associations.

Ontologies are broadly used in many domains to support user activities. Different use cases and tasks might require different visualisation focuses. It is hard to make a one-size-fits-all tool. One solution that supports some tasks properly might fail in others (Munzner, 2014; Dudáš et al., 2018). In the field of visualisation, the definition of *effectiveness* encompasses the match of both human cognitive system and intended use cases. It has been argued that the ontology visualisation needs to be designed around the user needs and the use cases that should be supported (Dudáš et al., 2018).

In this research, interviews have been conducted with experts in the biomedical domain to study user needs. The biomedicine domain dominates the ontology community, where more ontologies are developed and utilised than in most other domains (d’Aquin and Noy, 2012). From the interviews, the user activities with homogeneous and heterogeneous associations have been identified. The analyses related to homogeneous associations involve not only individual associations but more importantly strengths of associations between classes, investigated together with the underlying ontology hierarchy. The analyses performed with heterogeneous associations are more complicated. Thus, an effective ontology visualisation should allow users to easily spot the distribution of different types of associations in the ontology hierarchy and also support the exploration of association details. While most existing ontology tools focus on class hierarchy, they do not fully support these complex association analyses, requiring significant user effort.

While the use cases from the interviews in the domain of biomedicine motivated this research on visualising ontologies and associations, literature reviews have been conducted to explore other domains. Similar user activities with ontologies and associations have been identified in a number of application areas. For example, in the agronomy domain, many ontologies have been created to represent and analyse agronomic data (Jonquet et al., 2018; Drury et al., 2019). They are used to answer questions like “what are the appropriate rice varieties for a given soil or region?” and “how many rice varieties are bred from a particular breeding station?”. In the e-government domain, ontologies and

associations are involved in the analyses on citizens, authorities, or investment (Fraser et al., 2003; Wagner et al., 2006). For bibliometrics, the investigation of research areas, scientific collaborations, publication impact factor and grant funding relies on the support from ontologies, leading to the discovery of potential opportunities (Adam, 2002; Moed, 2006; Peroni and Shotton, 2018).

The limitations of existing tools and the demands and challenges for visualising large ontologies and associations motivate this research. Arising from these issues, the research objectives are formed, as discussed in the next section.

## 1.2 Research Objectives

The fundamental goal of this research is to investigate the facets of visualisations that can effectively support user needs for large ontologies and their associations. This goal can be broken down to answer the following research questions (RQ):

- RQ1: How can large ontologies be effectively visualised?

To improve the scalability of current ontology visualisations, an effective representation needs to be defined.

Visual compression techniques can be used to compress the elements that are not interesting. A measure of *interest* based on user needs and tasks is needed for data elements (detailed definition of *interest* will be discussed in Chapter 4). A compact and meaningful design needs to be produced to give a visual summary of the compressed elements.

Appropriate interaction techniques need to be defined to explore an ontology's large hierarchy and support user activities.

- RQ2: How can homogeneous associations be effectively visualised in an ontology?

An effective approach needs to be devised to visualise homogeneous associations alongside the ontology hierarchy structure. The associations should be clearly shown, including to which classes they apply.

The strength of associations between classes should be recognisable easily in the visualisation. The significant classes with most associations should stand out from other classes.

The interactivity should be well designed to support user activities with homogeneous associations and exploration.

- RQ3: How can heterogeneous associations be effectively visualised in an ontology?

A representation that can effectively visualise heterogeneous associations on the ontology hierarchy needs to be designed. Along with the demands for homogeneous associations, different types of associations should be distinguishable in the visualisation. The distribution of heterogeneous associations throughout the ontology hierarchy should be easily determined.

The interactivity should be well designed to support user activities with heterogeneous associations and exploration.

### 1.3 Research Methodology

This research makes use of the design process model proposed by Takeda et al. (1990), which has five sub-processes in the design cycle. *Awareness of the problem* refers to the exploration of the research field, to determine the problems under certain specifications. *Suggestion* process seeks the objectives to solve the problems. In the *Development* process, candidates for the solutions are developed. *Evaluation* is the evaluation of the candidates. *Conclusion* makes a decision of which candidate should be adopted.

Following this model, a user-centred design approach (Norman, 1988) was adopted, where users are involved in the initial needs gathering, visualisation design, prototype feedback and formal evaluation processes. Creative stimulations (Goodwin et al., 2013), such as presentations and demonstrations of alternative visualisations and tools, were involved to inspire users to think about the needs for visualisations and functionalities. Figure 1.3 shows the methodology of this research.

To understand user needs, interviews with domain expert and literature surveys are used as the starting point of this research, leading to the formulation of the research questions and research objectives. For each research question, the interviews identified details of the use cases and the issues with the current approaches, leading to visualisation objectives. Putting these objectives into practice, specific design requirements were derived. A prototype of the visualisation design was produced based on the requirements. For research question one and two, the prototype was evaluated through quantitative usability studies. The prototype was subsequently refined based on the results and feedback from the evaluation. Later, an expert user study with a similar structure to the prototype user study was conducted to test the usability of the refined prototype. The results and feedback from the experts were gathered and analysed, leading to further improvement to the visualisation. For research question three, the evaluation was based on domain expert feedback and walk-through case studies, since there is no comparable tool that satisfactorily supports the complex user activities on heterogeneous associations. Publications were produced to communicate the results.

### 1.4 Contributions

The main contributions of this research fall into three categories: 1) new representations of ontology hierarchies and associations, 2) design of ontology visualisation interactivity, and 3) implementation of the visualisation to support user activities.

Firstly, a number of new visual representations that meet the challenges posed by complex and large ontologies are presented:

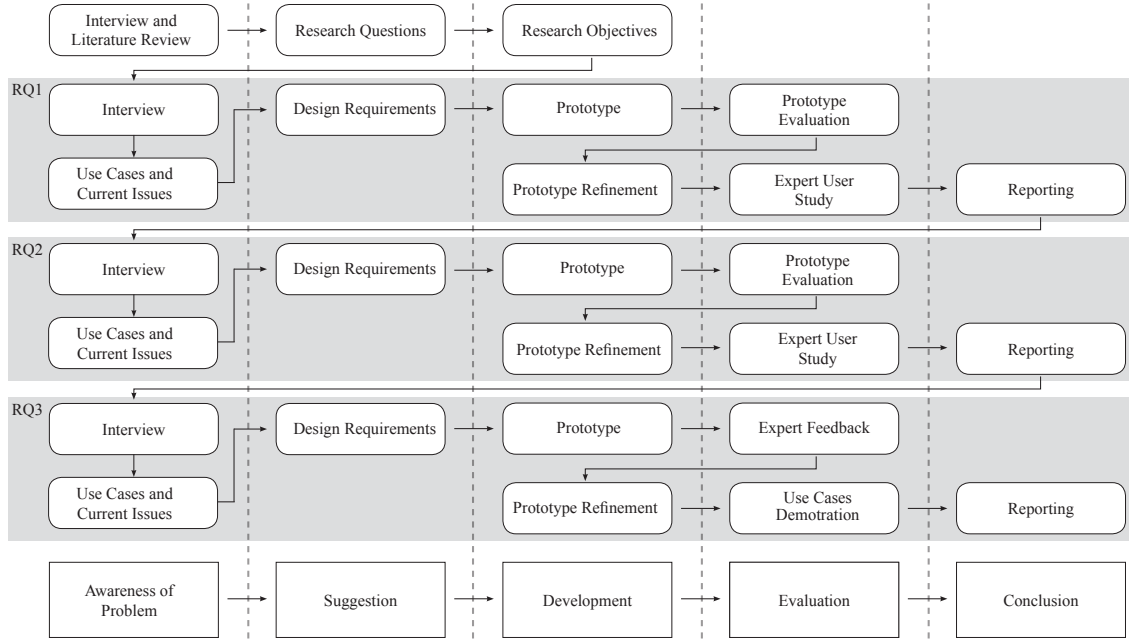


Figure 1.3: The user-centred research methodology for this research adapting the design process model (Takeda et al., 1990).

- a) A new visual representation, named OntoPlot, for visualising large ontology hierarchies is presented. It utilises the basic style of icicle plots but accommodates ontology classes in a more space-efficient way.
- b) Visual compression techniques are employed in the visualisation. An interest measure is defined to determine the data elements that need to be compressed, giving more focus on the interesting classes from the user's perspective.
- c) A set of distinct glyphs is designed to visually summarise the compressed structure, facilitating quick identification of different hierarchical structures.
- d) An effective approach to emphasising homogeneous ontology associations on top of the hierarchy is demonstrated. It is extended to show heterogeneous associations.
- e) A representation for showing the many-to-many heterogeneous associations between classes along with the ontology hierarchy is defined.

Second, a series of interactions are designed to support the exploration of ontology hierarchy and associations. Some particular features are highlighted below:

- a) The interest measure is dynamically calculated when users select properties or classes they are interested in, and the compression is automatically performed.
- b) The glyphs that represent compressed sections can be interactively expanded and collapsed. Animation is employed to ease the transition of changes.
- c) Appropriate features are utilised to ease the navigation around the ontology and preserve the context information for users.

Finally, as a realisation of the design, the browser-based software OntoPlot is developed. Its effectiveness to support user activities of ontology hierarchy and associations is demonstrated via the user studies and case studies. It is publicly available online, allowing users to visualise and interact with their own ontologies.

As discussed in Section 1.1, OntoPlot is domain agnostic and can facilitate equivalent ontology user activities for any domain. Also, while OntoPlot is designed to visualise ontologies and associations, it is generalisable to other hierarchically structured data, especially those containing additional non-hierarchy relations. One typical example is research collaborations between organisations, where the relationships between individuals in the hierarchy show how the researchers from different organisations work together, and the relationship strength indicates how often they collaborate.

## 1.5 Thesis Structure

This thesis is organised in an order the research was conducted, which progressively addressed each research question, building on the earlier work.

This introduction chapter briefly outlined the background of this research project, and presented the research motivations and contributions.

Chapter 2 introduces the background knowledge of ontology and ontology association data used in this research. It describes a user needs study via domain expert interviews. It reports the interviews for all three research questions, followed by the interview analysis to define use cases and design requirements for each research question.

Chapter 3 details the background of visualisation techniques related to this research. It discusses hierarchy visualisations and focuses on existing ontology visualisations, followed by the review of visual compression approaches.

Chapter 4 describes the design process for visualising large ontology hierarchies. It explores the design space and proposes the visual representation OntoPlot, with visual compression techniques and interactions. It addresses RQ1.

Chapter 5 presents the visualisation for homogeneous associations and describes the research process. The rest of the chapter presents evaluations and refinements. It addresses RQ2.

Chapter 6 explores and describes the approaches for visualising heterogeneous associations. It also demonstrates the usability of the system to support user activities through case studies. It addresses RQ3.

Chapter 7 concludes the research, as well as describes the directions for future research.



## Chapter 2

# User Needs

This chapter firstly gives a background of ontology and association data, then describes the process of analysing user needs for ontology and association visualisations. It presents interviews with a domain expert for understanding the work involved in ontology and association analysis. Next it describes the analysis of the interview data to identify use cases and visualisation design requirements for each research question. Some examples are also discussed in this chapter to show how domain experts currently perform the analyses using ontologies.

### 2.1 Ontology and Association

An ontology consists of a list of concepts and relationships between these concepts. Typically, a concept is a class of individuals, and their relationships are defined as binary relations (sometimes called *predicates*). The binary relations, expressed in the OWL (Horrocks et al., 2003) and OWL 2 languages (Cuenca-Grau et al., 2008), are usually defined as subsumption relations between classes.

The most prevalent relationships in an ontology are the subsumption relations between *named* classes, which denote *subclass* relations, indicating *inheritance*. One class  $C$  is stated as a *subclass* of another class  $C'$ , if every *instance* in  $C$  is included in  $C'$ . For example, in the Pizza ontology, `MushroomPizza` *subClassOf* `Pizza` presents a subclass relation. These inheritance relationships normally form the hierarchical structure for an ontology.

The inheritance relationships in an ontology are not required to form a strict hierarchy, meaning that a class may have multiple superclasses, denoting *multiple inheritance*. More specifically speaking, if a class  $A$  is a subclass of both  $B_1$  and  $B_2$ , then every instance of  $A$  is an instance of  $B_1$  and  $B_2$ .

Beside the *named* class subsumption relations, the subsumption relations can be expressed with properties (sometimes called *roles*) and *anonymous* classes. This kind of relationship places restrictions, such as *someValuesFrom*, *allValuesFrom* and *hasValue*, on properties. For example, in the relationship from the Pizza ontology `MushroomPizza` *subClassOf* `hasTopping some MushroomTopping`, the class `MushroomPizza` is linked to another class `MushroomTopping` through a subclass of a *someValuesFrom* restriction on the

property *hasTopping*. In this research, the term *association* is used to define this kind of restriction that involves a property.

Expressed in the OWL Description Logic syntax (Horrocks et al., 2003), the above two definitions can be formally expressed as follows. Axiom 2.1 states the subclass relationship (denoted ' $\sqsubseteq$ ') between named classes *MushroomPizza* and *Pizza*. Axiom 2.2 states the association on class *MushroomPizza*, asserting it as a subclass of a *someValuesFrom* value restriction.

$$\text{MushroomPizza} \sqsubseteq \text{Pizza} \quad (2.1)$$

$$\text{MushroomPizza} \sqsubseteq \exists \text{ hasTopping.MushroomTopping} \quad (2.2)$$

As introduced in Chapter 1, *homogeneous* and *heterogeneous* associations here refer to a set of associations involving the same or different properties respectively. There may also be inheritance relations between properties in an ontology, forming a property hierarchy, which enriches the relationships between heterogeneous associations.

In addition, properties can have different facets, such as *cardinality* and *value type* (Noy et al., 2001). Cardinality restrictions define the number of values a property can have. Value type describes the types of values for a property, such as string, number or dates, which are atomic values also called *literals* (Antoniou et al., 2012).

To enhance the expressiveness of ontologies, besides the binary relations, there are other definitions that are used in ontology languages to express the relationships between classes, such as *equivalence* and *disjointness* (Antoniou et al., 2012). If two classes have exactly the same instances, they are equivalent. If two classes do not have any common instances, they are disjoint. This kind of ontology expressivity enriches the semantics of ontologies and facilitates complex knowledge management (Antoniou et al., 2012).

The targets of this research are the *homogeneous* and *heterogeneous* associations asserted by *someValuesFrom* restriction. Other axioms involving *allValuesFrom*, *hasValue*, *cardinality*, *literals*, *equivalence* or *disjointness* are not the focus of association analyses, so will not be dealt with.

### 2.1.1 Biomedical Ontologies

Ontologies have been widely adopted for the purpose of knowledge representation in a number of areas, for example, biology, medicine, agronomy, e-government and bibliometrics, etc. (d'Aquin and Noy, 2012). Biomedical research is one of the popular and most successful applications of ontologies and it dominates the ontology community (Stevens and Lord, 2009; Shah and Musen, 2009; d'Aquin and Noy, 2012). Thus, biomedical ontologies and their use is the focus of this research.

In biomedical research, there is an abundance of heterogeneous data, including genes, proteins, clinical observations and laboratory data, that need to be integrated to facilitate the formulation, evaluation and refinement of hypotheses. Biomedical ontologies drive the computational use of this biological data. They achieve this by organising and classifying

knowledge in a formalised and structured manner, providing unambiguous and shareable descriptions (Shah and Musen, 2009).

Biomedical ontologies provide a basis for integrating and understanding knowledge from multiple sources. The shared understanding of collected data is essential for biologists to describe the same entities in the same way. One typical example is the widely-used Gene Ontology (Ashburner et al., 2000), which defines 50,000 classes to annotate biological entities (i.e., genes and gene products) that result from high-throughput experiments.

Besides the use of ontology itself, ontology associations also serve as a solution provider in the biomedical research community. The structure of associations between classes in an ontology can be used to query the data and to explore how the data relate to each other, allowing biologists to perform the analyses related to ontology relationships.

The structure of associations in biomedical ontologies is diverse and complex. These associations can be one-to-one (e.g., a class linked to another class) or many-to-many (e.g., some classes linked to many other classes). They also can be of the same type (i.e., homogeneous associations, involving one property) or of different types (i.e., heterogeneous associations, involving multiple properties).

To better understand user needs for ontologies and associations, interviews were conducted with a domain expert in bioinformatics, which will be discussed in detail in Section 2.2.

### 2.1.2 Multiple Ontologies

Although this research explores the use of single ontologies and the associations included within them, it is worth giving a brief overview of the use of multiple ontologies. Linking classes from multiple ontologies is another important area of research in the ontology community, which is widely used to support ontology mapping, ontology term reuse and ontology evolution investigation.

Ontology mapping is defined as a specification of the semantic overlap between two ontologies (Kalfoglou and Schorlemmer, 2003). It is seen as connection between different ontologies. As the number of ontologies increases significantly, there is a need to connect ontologies to support interoperability (Falconer and Storey, 2007). Mappings are then provided as translation rules to resolve differences in terminology, syntax or language between ontologies. There are five types of mappings: lexical mapping (simple lexical comparison between term labels), *xref* mapping (a term is referred to another term by analogousness or other predicates), CUI mapping (terms assigned with the same Concept Unique Identifier), URI mapping (terms assigned with the same Uniform Resource Identifier) and user submitted mapping. Mappings among ontologies constitute a key component that enables the use of the ontologies for data integration and information exchange (Flouris et al., 2008). Understanding ontology mappings and how ontologies are related is a critical step in integrating data and using multiple ontologies at the same time in applications.

Ontology term reuse refers to the situation for ontology development where the same term is present in two or more ontologies either by explicit reference directly using the same IRI (Internationalised Resource Identifier), or via *xref* (a term is referred to another

term by analogousness or other predicates) or CUI (Concept Unique Identifier). Ontology development is seen as a reuse-oriented process (Noy et al., 2001). Reusing terms from existing ontologies is encouraged by most ontology development methodologies to create accurate and cost-effective ontologies and avoid redundancy of classes. However, according to Kamdar et al. (2015) who conducted an investigation of term reuse for the ontologies in BioPortal, term reuse is estimated to be rather low, less than 5%, while term overlaps between ontologies is as high as about 14%. Term overlap refers to the situation in which two terms are similar by their labels or synonyms, but do not have any reference to each other. In other words, overlapping terms in different ontologies are regarded as redundant. The set of terms resulting from subtracting term reuse from term overlap is called the *overlap-reuse gap*. To minimise this gap, the overlapping terms between different ontologies should be identified and replaced by reuse of existing terms.

To support the evolution of biological science and the change of knowledge, and to correct errors and logical inconsistency, ontologies are updated continuously. In this case, different versions of an ontology are generated. Gene Ontology (GO) can be taken as an example. GO is updated daily (Rhee et al., 2008). All the changes are tracked to allow versioning of the ontology. According to Dameron et al. (2013), between January 2008 and December 2012, the number of classes in GO increased by 50%, and the number of relations grew by 85%. Meanwhile, the hierarchical structure of GO also changed. GO consists of three branches: Biological Process (BP), Cellular Component (CC) and Molecular Function (MF). The new classes were mostly added to the intermediate levels rather than leaves for BP, whereas classes are added mostly as leaves for CC and MF. Given such changeable information, it is important for a GO user to understand the similarity and difference of various versions of GO and their structures, in order to accurately describe the biological reality (Hill et al., 2008).

The use of multiple ontologies and the user needs for their visualisations are not addressed specifically in this research. However, more effective visualisation of single ontologies can serve as a foundation for building multiple ontology visualisation systems.

## 2.2 Exploratory Interviews

The data of interest in this research are ontologies and their contained associations as introduced in Section 2.1.1. To better understand the analytical activities with such data, a series of interviews was conducted with an experienced domain expert in bioinformatics, whose research focuses on the development and application of biomedical ontologies, and who has actively published in these areas for fifteen years. In order to address the different research questions, the domain expert was interviewed a number of times over the whole period of this research.

The interviews were semi-structured, where a list of prepared questions was used to prompt the expert and follow-up questions allowed deeper discussion (Williamson and Johanson, 2017). The prepared questions focused on the high-level intentions of the activities, shown as follows:

- What is the background of this work?
- What is the motivation of this work?
- What is the use case for this analysis?
- How do you perform the analysis currently?
- Did you encounter any difficulties?
- What features of the tool that you are using do you like or dislike?
- What new functions would you expect?

The interview data was gathered and analysed to retrieve use cases. These use cases were broken down using the analysis framework by Munzner (2014) firstly to abstract low-level actions and targets from each use case, then to identify the similar operations from the use cases and provide structured guidelines to construct visualisation design requirements. Besides the discussion with the expert, the literature that describes related works was studied to get a thorough understanding of the current approaches. The outcome of these analyses is described in the following sections, organised by research question.

Although the expert involved in the interviews works in the bioinformatics domain, the identified use cases are not described in the context of biology or biomedicine and are applicable to other ontology application areas, as discussed in Chapter 1.

## 2.3 Large Ontologies

This section describes the use cases, visualisation requirements and current approaches that relate to RQ1: how can large ontologies be effectively visualised?

### 2.3.1 Use Cases

When working with large ontologies, the preliminary focus is the content of the ontology and the topology of the ontology hierarchy. During early interviews with the domain expert, a number of common use cases that are important for using large ontologies were identified. The actions and targets derived from each use case are summarised in Table 2.1.

**U1 Describe data.** Given an ontology, experts want to be able to find classes in the ontology and inspect the class information. This is important to understand and describe the content of the ontology. Experts also want class labels to be shown so they can easily spot classes.

**U2 Focus on *interesting* information.** As ontologies can be very large and can contain rich information, experts want to be able to focus on the required information. The required information is considered to be the *interesting* parts of an ontology. Note, the parts of interest might be different for different users.

**U3 Generalise concepts.** While investigating an interesting class in an ontology, experts want to know the path from the class to the root, being able to generalise the class concept from the concepts of its ancestors. For example in Figure 2.1, class A is the root.  $F \rightarrow E \rightarrow A$  is the path from class F to the root.

**U4 Discover common knowledge.** Given classes of interest, experts want to be able to easily trace their paths towards the root and determine the concept that is the lowest common ancestor, facilitating the discovery of common knowledge. For example in Figure 2.1, both A and B are the common ancestors of C and D, while B is the lowest common ancestor of C and D.

**U5 Discover new knowledge.** In order to discover new knowledge beyond the *interesting* parts of an ontology, experts will also sometimes need to access and browse the entire ontology to explore the contained information.

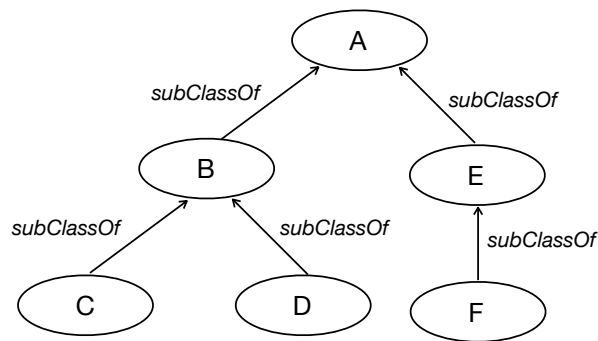


Figure 2.1: A simple illustration hierarchy.

Table 2.1: Common use cases when working with ontologies.

Use Case	Description	Need	Action	Target
U1	Describe data	Find class and its information	Search	Class
			Inspect detail	Class
		See class label	Locate	Class
U2	Focus on <i>interesting</i> information	Easily spot required information	Define	Class of interest
			Locate	Class of interest
U3	Generalise concepts	See the path from a class to the root	Identify	Topology
U4	Discover common knowledge	Find the lowest level of common ancestors for particular classes	Identify	Topology
U5	Discover new knowledge	Access the entire ontology	Explore	Ontology
			Navigate	Ontology

### 2.3.2 Visualisation Requirements

From the use cases and their actions and targets enumerated above, along with the typical nature of ontological data, a number of specific design requirements for an interactive system to explore large ontologies are identified (see also Table 2.2).

- R1** In order to present the content of an ontology (U1), the visualisation should support the access of class information contained in the ontology and provide a search function to allow users to easily find useful information.
- R2** To better describe ontologies (U1), the visualisation should be expressive, which is able to display the labels of classes.
- R3** In order to let users focus on the *interesting* parts of an ontology (U2), the visualisation should give more prominence to these parts to allow them stand out from the ontology hierarchy and be easily spotted by users. Note, the interesting parts may be spread across the whole ontology.
- R4** Ontologies encode a clear hierarchical structure through subclass relationships and the hierarchy is the most useful way to arrange large ontologies. Also, the hierarchical structure is essential for U3 and U4 to find the path from a class to the root, so it must be prominently represented.
- R5** As described in U5, the entire ontology should be embodied in the visualisation. Since ontologies can be very large, the visualisation is required to maximise the use of available space. Since ontologies are generally broad (i.e., much wider than they are deep), with traditional hierarchy visualisations, the branches with large numbers of leaf nodes (the nodes without any children, for example C, D and E in Figure 2.1) will take up significant amounts of horizontal space. There is a need to give less prominence to uninteresting branches with large numbers of leaf nodes.
- R6** For U5, the visualisation should support the exploration of the ontology, allowing users to easily browse and navigate through the different parts of the ontology.

Table 2.2: Visualisation design requirements for ontologies in the context of identified use cases.

Requirement	Description	Use Case
R1	Provide class information and search function.	U1
R2	Show class labels.	U1
R3	Emphasise <i>interesting</i> parts.	U2
R4	Clearly present ontology hierarchy structure.	U3, U4
R5	Maximise the use of available screen space.	U5
R6	Support browsing through the entire ontology.	U5

### 2.3.3 Current Approach

The tool used by domain experts to perform analyses on ontologies is Protégé (Noy et al., 2000). Protégé is the most widely used ontology editor. It offers an indented list, which is similar to file browsers (see Chapter 3 for a detailed description of indented list), to represent the ontology hierarchy, annotating the classes with their labels (U1) (see Figure 2.2 ①). It presents the class information as text lists in separate views (U1) (see Figure 2.2 ②) and provides a search window (U1) (see Figure 2.2 ③). Users can click a class to expand or collapse a subtree to show or hide the classes based on their interests (U2). The indented list can be scrolled down to browse the entire ontology (U5).

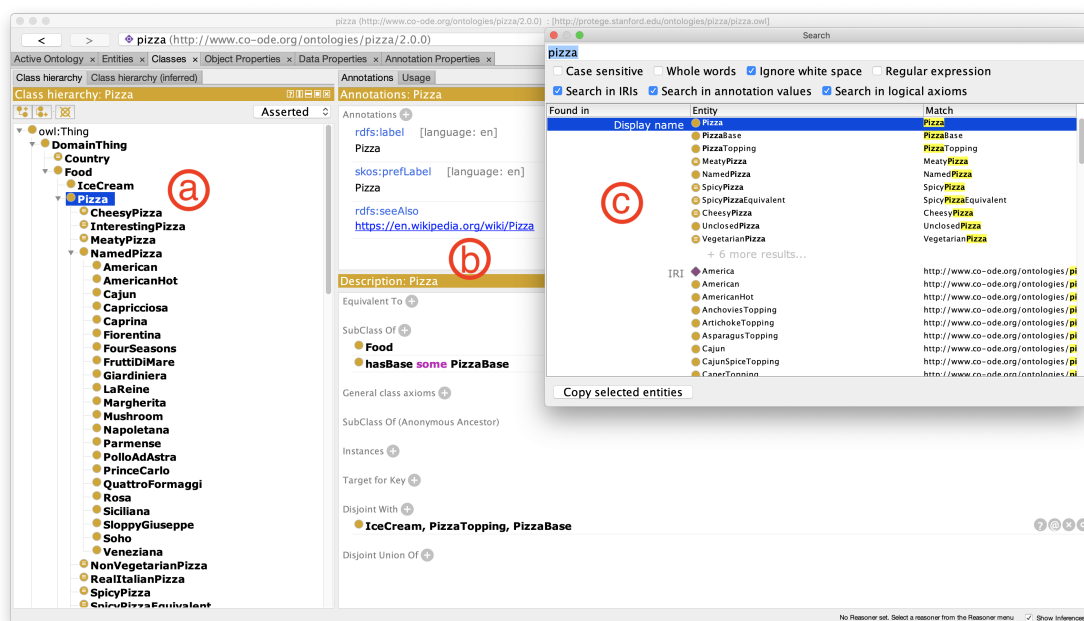


Figure 2.2: The interface of Protégé, visualising the Pizza ontology.

The expert commented that using Protégé to trace the path in the ontology hierarchy (U3) is not easy, especially when the class is far away from the root in a very deep or wide ontology. Also, finding common ancestors for classes (U4) is not well supported by Protégé.

## 2.4 Homogeneous Associations

This section presents the use cases, visualisation requirements and current approaches that relate to RQ2: how can homogeneous associations be effectively visualised in an ontology?

### 2.4.1 Use Cases

During the interviews with the domain expert, the process of using biomedical ontologies for exploring and cataloguing the homogeneous associations was discussed. This type of work involves understanding not only the underlying hierarchy, but also strengths of associations between classes in different parts of the ontology. A number of common use



cases that are important for such work are identified. The actions and targets derived from each use case are summarised in Table 2.3.

**U6 Inspect a class’s associations.** For a class of interest, experts want to see the distribution of its associations across the ontology hierarchy. Experts also want to know the number of associations and the association details.

**U7 Detect significant associations.** Experts strongly demand the need to easily identify the classes with the greatest strength of associations from the ontology.

**U8 Identify class effect.** Experts expect to be able to clearly identify when particular types of associations apply to most or all child classes of a given class, i.e., the associations’ effect on a class of things.

**U9 Predict possible associations.** Experts want to be able to explore the siblings of classes (e.g., in Figure 2.1 C is the sibling of D) with a given association, since a sibling that does not have the association might be suspected of having the association and be tested for it.

Table 2.3: Common use cases when working with homogeneous associations in ontologies.

Use Case	Description	Need	Action	Target
U6	Explore a class’s associations	See the distribution of associations See the details of associations	Locate Inspect detail	Association Association detail
U7	Detect significant associations	Identify the classes with the greatest strength of associations	Locate Inspect	Association Association strength
U8	Identify class effect	See when associations apply to a number of child classes	Locate Identify	Association Topology
U9	Predict possible associations	Show the sibling of classes with associations	Locate Explore Identify	Association Ontology Topology

### 2.4.2 Visualisation Requirements

Through the interviews with the domain expert, the process of analysing homogeneous associations was discussed in detail. In response to the use cases involved in the process and their actions and targets that are described above, the design requirements of the visualisation are defined (see also Table 2.4).

**R7** When considering associations in large ontologies, those associations might only apply to a small subset of the ontology. An effective visualisation needs to clearly highlight the parts of the ontology with relevant associations (U6, U8, U9) and emphasise those with the greatest strength (U7).

- R8** The visualisation should allow users easily to obtain the detailed information of associations (U6).
- R9** Where there are large parts of the ontology without relevant associations, the visualisation should be able to hide or show these so that users can consider just the relevant parts of the ontology, or optionally view the ontology in its entirety as desired (U9).
- R10** When associations are related to children of a particular class (U8), then the class effect needs to be easily identified.

Table 2.4: Visualisation design requirements for homogeneous associations in the context of identified use cases.

Requirement	Description	Use Case
R7	Clearly highlight the parts of the ontology with (significant) associations.	U6, U7, U8, U9
R8	Show association details and strength.	U6, U7
R9	Hide or show the parts of the ontology without associations.	U9
R10	Clearly highlight class effects.	U8

### 2.4.3 Current Approach

Again, Protégé is the primary tool used by the domain expert to analyse homogeneous associations. As discussed in Section 2.3.3, the association information of each class is listed as text in the “Class Description” view (U6) (see Figure 2.2 ⑤). In order to locate each associated class of a particular class in the ontology hierarchy (U6), experts must click each association class label in the “Class Description” view, which the expert described as a “tedious process”.

Then experts mark the association classes in the ontology hierarchy to investigate their ancestors and siblings (U8, U9). Figure 2.3 shows an example of how experts present the association classes for the **Fatty Acids [Chemical/Ingredient]** class in the ontology hierarchy to find the class effect (U8). The ontology hierarchies used here are the screenshots of the Protégé indented list, with manual annotations added in red.

To figure out the number of associations for each class (U7), domain experts either count the associations manually or write SPARQL queries to do this. SPARQL is a semantic query language developed to retrieve and manipulate RDF (Resource Description Framework) format graphs such as ontology data (Prud et al., 2008). Figure 2.4 shows one of the SPARQL queries written by experts for this purpose.

The domain expert commented that learning to write SPARQL queries is time consuming. Also, writing a valid query to get the correct result is not an easy task, especially when the query syntax is complicated, consisting of various operations and restrictions.

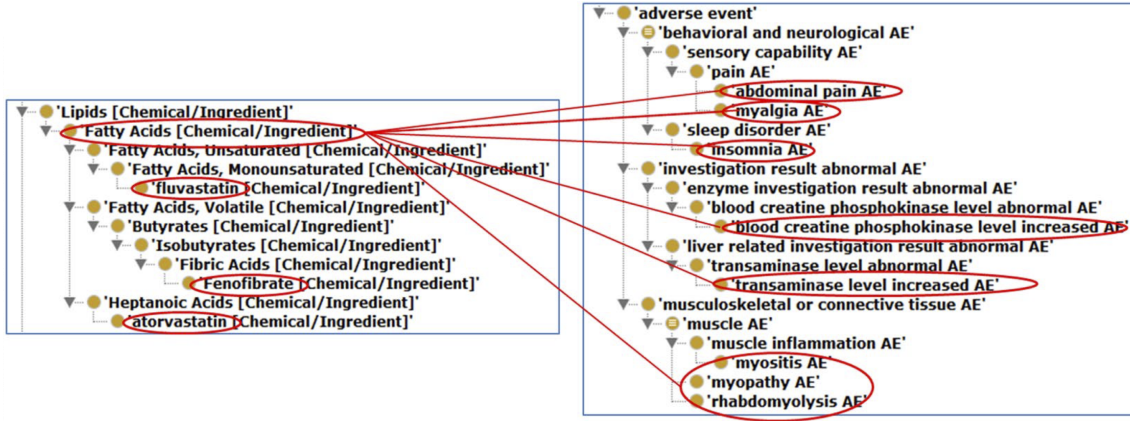


Figure 2.3: An example of how domain experts manually annotate screenshots of the Protégé indented list to show the class effect: the Fatty Acids [Chemical/Ingredient] class (left) has association with all the leaf classes in the adverse event subtree (right) (Wang et al., 2017).

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
prefix obo: <http://purl.obolibrary.org/obo/>

SELECT ?p_dae ?labelp (COUNT(*) AS ?count)
WHERE {
  SELECT ?p_dae ?labelp ?dae ?label1 ?drug ?labeldrug ?s1
  WHERE {
    ?dae rdfs:subClassOf ?s1 .
    ?dae rdfs:subClassOf ?p_dae .
    ?p_dae a owl:Class .
    ?p_dae rdfs:label ?labelp .
    ?dae rdfs:label ?label1 .
    ?s1 owl:onProperty obo:BF0_0000057; owl:someValuesFrom ?drug .
    ?drug rdfs:label ?labeldrug .
    FILTER REGEX(str(?p_dae), "OAE") .
  }
} GROUP BY ?p_dae ?labelp

```

Figure 2.4: An example of the SPARQL query written by domain experts (Guo et al., 2016).

To present the number of associations for classes in the ontology hierarchy and to identify the class having greatest association strength (U7), experts usually add annotations manually to the screenshots of the Protégé indented list to indicate the association strength (see Figure 2.5).

Figure 2.6 shows another example of the visualisation used by experts to present the association strength for classes (U7). In this figure, the bubble chart in view ① was generated by Tableau. In this view, each class is represented as a circle. The size and colour saturation of each circle represents the association strength. The bigger size and more vivid colour indicate greater strength. Only the classes crossing the strength threshold are labelled. The ontology hierarchy is presented in view ②, using the screenshot of the Protégé indented list. The classes that are able to be labelled in view ① are annotated with asterisks in this hierarchy.

In this visualisation, even with the labels in the ontology hierarchy, it is still hard to visually match the same classes in the bubble chart and in the hierarchy.

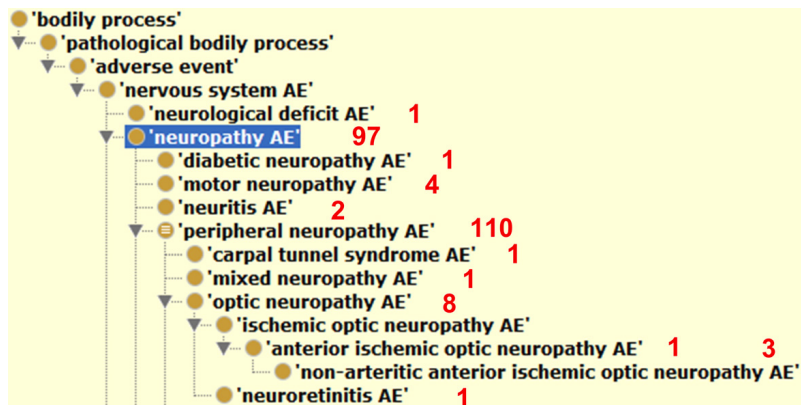


Figure 2.5: An example of the manual approach to show association strength within an ontology: association numbers written next to branches of the hierarchy on a screenshot of Protégé indented list view (Guo et al., 2016).

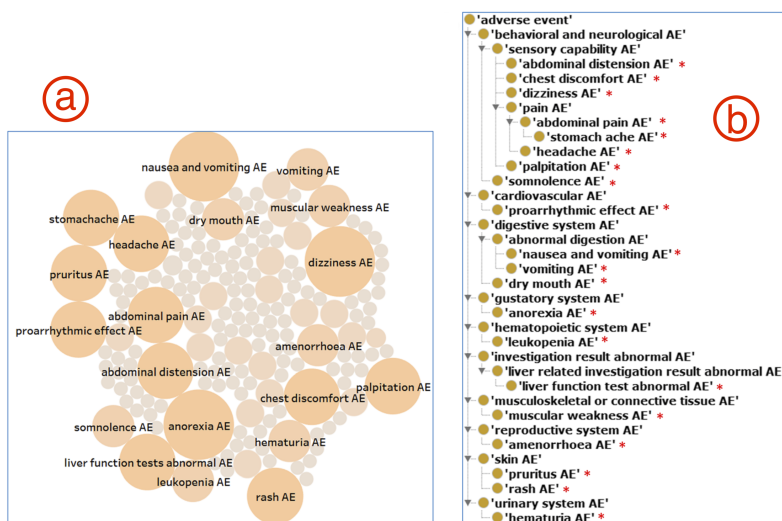


Figure 2.6: A bubble chart alongside an annotated Protégé indented list was used by domain experts to show the association strength for the ontology classes (Liu et al., 2017).

## 2.5 Heterogenous Associations

This section describes the use cases, visualisation requirements and current approaches that relate to RQ3: how can heterogeneous associations be effectively visualised in an ontology?

### 2.5.1 Use Cases

Another major use of ontology associations is related to heterogeneous associations. The general analyses that domain experts perform involve either comparing different types of associations or linking associations to explore relationships between classes. The detailed use cases that are identified from the interviews with the expert are listed below. Their derived actions and targets are summarised in Table 2.5. Each use case is described in terms of the simple example ontology depicted in Figure 2.7.

- U10 Compare heterogeneous associations.** Experts want to see the various distributions of the different types of associations in the ontology hierarchy and the details of the classes they are applied to. For example in Figure 2.7, comparisons can be performed between the heterogeneous associations involving *hasBase* and *hasTopping*.
- U11 Identify intersection classes of heterogeneous associations.** When comparing different types of associations, experts are also looking for the intersection classes that are involved in all the interesting types of associations. For example in Figure 2.7, *MushroomPizza* is the intersection class of the heterogeneous associations involving *hasBase* and *hasTopping*.
- U12 Compare heterogeneous associations for individual classes.** There is a need to compare heterogeneous associations at class level. Experts want to see the different distributions of various types of associations on interesting classes. For example in Figure 2.7, experts want to compare the different distributions of the heterogeneous association involving *hasBase* and *hasTopping* on the classes *MushroomPizza* and *ItalianPizza*.
- U13 Link heterogeneous associations.** The relationships between classes in an ontology can be built and linked through heterogeneous associations. When performing analysis on this kind of relationship, experts want to investigate the different types of associations together and the classes involved in these associations. For example in the ontology shown in Figure 2.7, *MushroomPizza* can be linked to *Mushroom* via the class *MushroomTopping* and two properties *hasTopping* and *hasSauce*, from which they can infer that *MushroomPizza* has *Mushroom*.
- U14 Identify commonly associated classes for classes.** When performing analyses comparing or linking heterogeneous associations, experts need to inspect particular interesting classes and identify the common classes associated with these classes. The types of associations linking these classes should also be visualised. For example in Figure 2.7, given the heterogeneous associations involving *hasSauce* and *hasFried*, experts want to be able to identify *Mushroom* is the commonly associated class of *MushroomTopping* and *MushroomSoup* by both properties.

### 2.5.2 Visualisation Requirements

As can be seen from the above use cases and their actions and targets, the analyses on heterogeneous associations involve more data elements and greater complexity than homogeneous association analyses. Thus, the visualisation will require more sophisticated design consideration to support these analytical activities to investigate heterogeneous associations. Such design requirements that are derived from the use cases and the actions and targets are described below (see also Table 2.6).

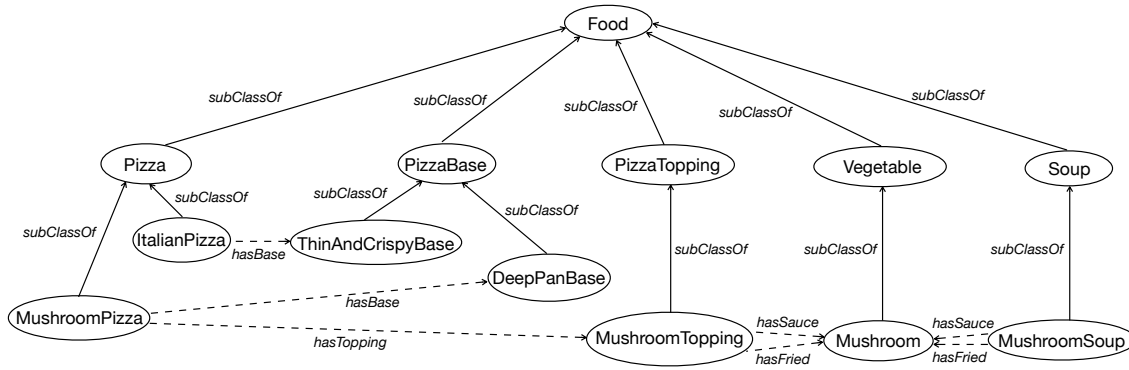


Figure 2.7: An illustration ontology adapting the Pizza ontology to illustrate simple heterogeneous association use cases.

Table 2.5: Common use cases when working with heterogeneous associations of ontologies.

Use Case	Description	Need	Action	Target
U10	Compare heterogeneous associations	See the distribution of associations See the details of involved classes	Locate Inspect detail	Heterogeneous associations Class
U11	Identify intersection classes of heterogeneous associations	Easily spot the intersection classes	Locate	Class
U12	Compare heterogeneous associations for individual classes	See the distribution of associations on classes	Locate Inspect	Heterogeneous associations Multiple classes
U13	Link heterogeneous associations	See the distribution of associations on classes	Locate Inspect	Heterogeneous associations Multiple classes
U14	Identify commonly associated classes for classes	Easily spot the common classes	Locate Locate Inspect	Class Heterogeneous associations Multiple classes

**R11** As required in U10, the visualisation should be able to represent different types of associations in the ontology hierarchy, allowing users easily spot their distribution on the ontology classes.

**R12** In order to give the information about the classes involved in the heterogeneous associations (U10), the visualisation should be able to display the class details.

**R13** To reveal the intersection classes of all the interesting types of associations (U11), the visualisation should be able to automatically highlight the classes that have these types of associations and filter out the other classes.

- R14** To support U12, U13 and U14, the visualisation should be able to display the interesting heterogeneous associations and clearly show how these associations are distributed among multiple classes.
- R15** For the interesting classes, the visualisation should allow multiple selections of these classes and automatically highlight the common classes that associate with all of them and filter out the other classes (U14).

Table 2.6: Visualisation design requirements for heterogeneous associations in the context of identified use cases.

Requirement	Description	Use Case
R11	Clearly show the distribution of heterogeneous associations in hierarchy.	U10
R12	Show class details.	U10
R13	Detect and display intersection classes for particular association types.	U11
R14	Clearly show the distribution of heterogeneous associations on multiple particular classes.	U12, U13, U14
R15	Detect and display the classes commonly associate with multiple particular classes.	U14

### 2.5.3 Current Approach

When analysing heterogeneous associations, domain experts also use Protégé to investigate the associations for classes. Figure 2.8 shows an example where experts find three interesting associations (the blue highlighted item in view ⑥) for the FLUVASTATIN NA 20MG CAP [VA Product] class (the blue highlighted item in view ⑤) that are linked by three different properties (U13). In this screenshot of the Protégé indented list (see view ⑤), experts manually highlight the three classes (outlined with blue, green and red respectively) that are associated with FLUVASTATIN NA 20MG CAP [VA Product]. This is done manually because there is no possibility in Protégé to highlight more than one class on the hierarchy at the same time. From these three associations, experts then identify that the fluvastatin [Chemical/Ingredient] entity is associated with paresis AE, linked through the FLUVASTATIN NA 20MG CAP [VA Product] drug.

Figure 2.9 is a heatmap generated in R to show the association strength between adverse events and molecular entities under different groups of drug products (U12, U14). The expert commented that although the heatmap can show the association strength between different adverse events and molecular entities clearly, it is missing the hierarchical context for the classes.

To compare heterogeneous associations, experts manually check the classes that have interesting associations and annotate the properties on the screenshots of the Protégé indented list (U10, U11). For example, in Figure 2.10a, the annotation a-i represents anti-inflammatory and a-n stands for antineoplastic. Similarly, in Figure 2.10b, the annotations G, O, R represent human gut, human oral cavity, human respiratory airway respectively.

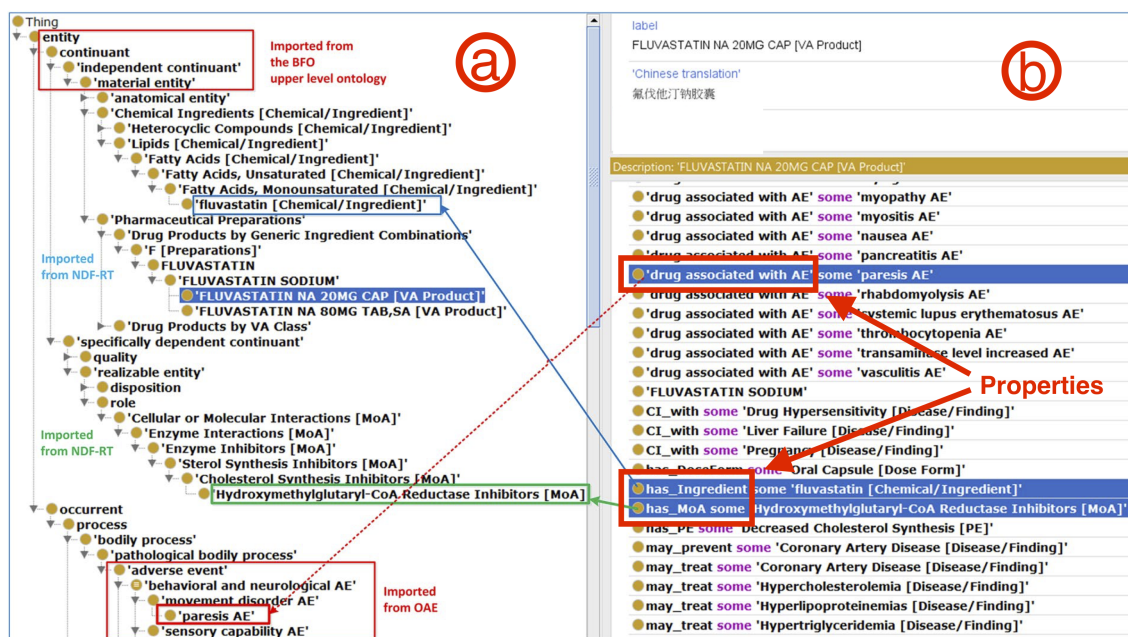


Figure 2.8: An example of how domain experts present three heterogeneous associations in the ontology hierarchy by annotating screenshots of the Protégé indented list view (Wang et al., 2017). ①: the indented list view of ontology hierarchy, ②: the class description view listing associations for the selected class FLUVASTATIN NA 20MG CAP [VA Product].

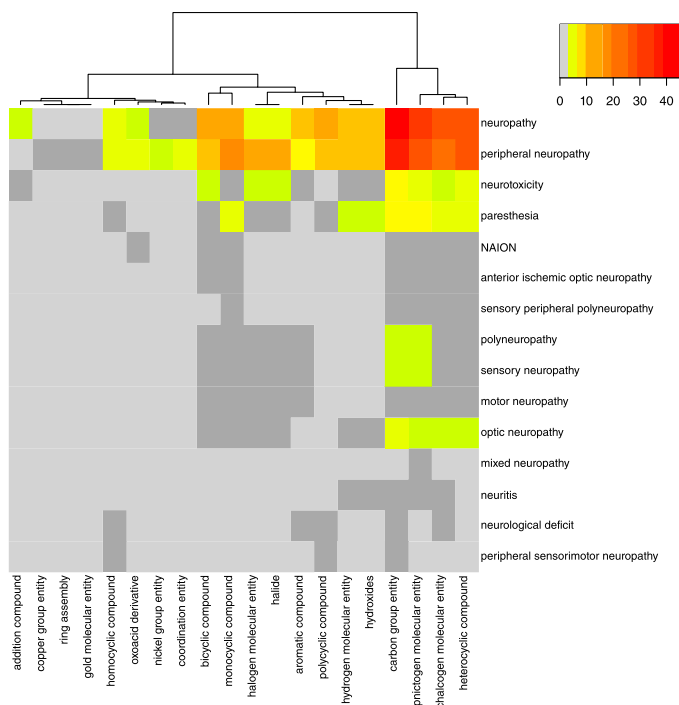
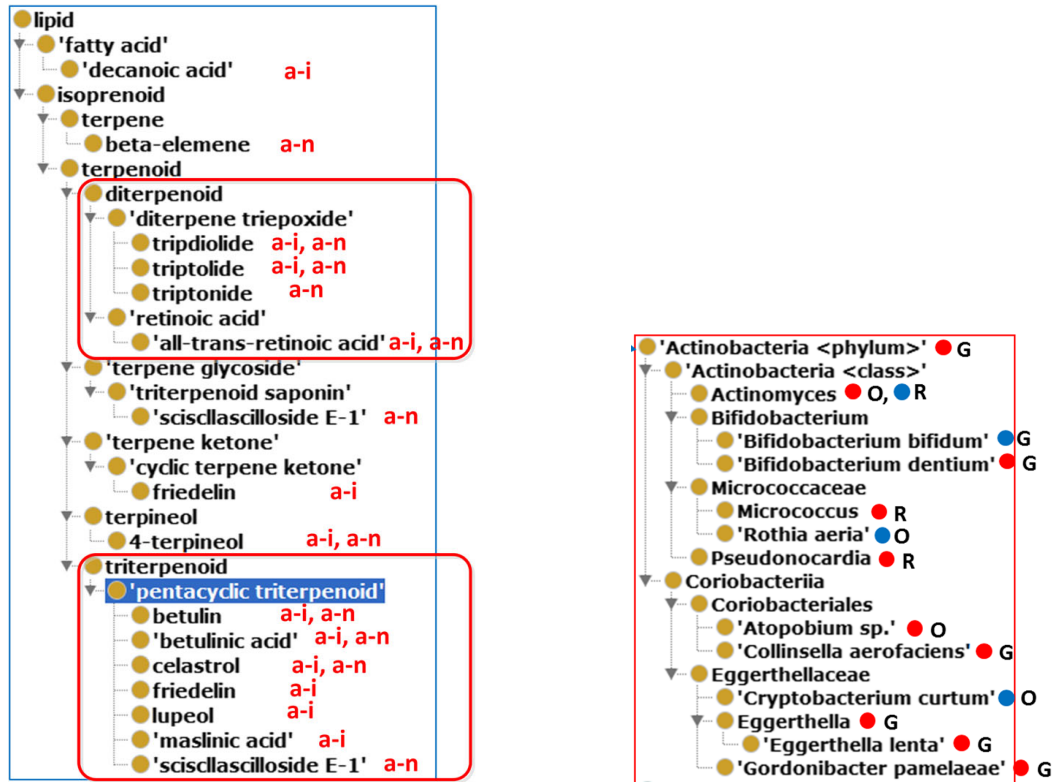


Figure 2.9: The heatmap used by experts to show the association strength between adverse events and molecular entities linked by drug products (Guo et al., 2016).



Here, the circles with different colours are used to indicate different activities, with red representing increased profile and blue representing decreased profile. Experts use this kind of visualisation to show the distributions of different types of associations. Such a visualisation will be less effective when the number of association types increase, as it will become hard to distinguish different properties. Also, it cannot be used to identify which classes are associated with a class.



(a) Different properties are annotated by different strings. Two types of associations (a-i and a-n) are shown (Liu et al., 2017).

(b) Different properties are annotated by a combination of different symbols and characters. Six types of associations (increasing and decreasing of G, O, R) are shown (He et al., 2019).

Figure 2.10: Examples of using screenshots of Protégé indented list to show heterogeneous associations on ontology hierarchies.

## 2.6 Summary

Table 2.7 summarises the use cases, visualisation requirements and current approaches described for the research questions in the above sections, derived from the interviews with the domain expert.

As can be seen from the table, performing the analyses that involve ontology hierarchy and complex associations currently requires considerable manual effort. There are strong needs for designing visualisations to support such use cases.

Table 2.7: Summary of the interview data analysis outcome.

Use Case	Visualisation Requirement	Action	Target	Feasibility of Current Approach
U1 Describe data	Provide class information and search function	Inspect detail	Class	Possible
		Search	Class	Possible
	Show class labels	Locate	Class	Possible
U2 Focus on <i>interesting</i> information	Emphasise <i>interesting</i> parts	Define	Class of interest	Possible
		Locate	Class of interest	Very Difficult
U3 Generalise concepts	Clearly present ontology hierarchy structure	Identify	Topology	Difficult
U4 Discover common knowledge	Clearly present ontology hierarchy structure	Identify	Topology	Difficult
U5 Discover new knowledge	Maximise the use of available screen space			
	Support browse through the entire ontology	Explore Navigate	Ontology Ontology	Possible Possible
U6 Explore a class's associations	Clearly highlight the parts of the ontology with associations	Locate	Association	Very Difficult
	Show association details and strength	Inspect detail	Association	Very Difficult
U7 Detect significant associations	Clearly highlight the parts of the ontology with significant associations	Locate	Association	Very Difficult
	Show association strength	Inspect	Association strength	Very Difficult
U8 Identify class effect	Clearly highlight the parts of the ontology with associations	Locate	Association	Very Difficult
	Clearly highlight class effects	Identify	Topology	Difficult
U9 Predict possible associations	Clearly highlight the parts of the ontology with associations Hide or show the parts of ontology without associations	Locate	Association	Very Difficult
		Explore	Ontology	Possible
		Identify	Topology	Difficult

Table 2.7: Summary of the interview data analysis outcome continued.

Use Case	Visualisation Requirement	Action	Target	Feasibility of Current Approach
U10 Compare heterogeneous associations	Clearly show the distribution of heterogeneous associations in hierarchy	Locate	Heterogeneous associations	Very Difficult
	Show class details	Inspect detail	Class	Possible
U11 Identify intersection classes of heterogeneous associations	Detect and display intersection classes for particular association types	Locate	Class	Very Difficult
U12 Compare heterogeneous associations for individual classes	Clearly show the distribution of heterogeneous associations on particular classes	Locate	Heterogeneous associations	Very Difficult
		Inspect	Multiple classes	Very Difficult
U13 Link heterogeneous associations	Clearly show the distribution of heterogeneous associations on particular classes	Locate	Heterogeneous associations	Very Difficult
		Inspect	Multiple classes	Very Difficult
U14 Identify commonly associated classes for classes	Detect and display the classes commonly associated with particular classes	Locate	Class	Very Difficult
	Clear show the distribution of heterogeneous associations on particular classes	Locate	Heterogeneous associations	Very Difficult
		Inspect	Multiple classes	Very Difficult

## 2.7 Conclusion

Ontologies and associations have been widely used in many domains. They facilitate the analysis of relationship semantics in ontologies. Visualisation systems should have targets and be able to support specific use cases. Thus, user needs analysis is an essential process before designing a visualisation. This chapter discussed the interviews with a domain expert and the subsequent interview data analyses for understanding user needs of ontology and association visualisations.

The next chapter will explore existing visualisations and evaluate their effectiveness. The outcome of the user needs analysis in this chapter and the literature review in the next chapter motivates and guides the visualisation design and evaluation that will be discussed in the later chapters for answering the three research questions.

## Chapter 3

# Visualisation Techniques

This chapter discusses the background of visualisation research related to this thesis. It begins with an overview of hierarchy visualisations. Then the discussion focuses on ontology visualisations and ontology association visualisations. Following this, there is a review of visual compression approaches.

### 3.1 Hierarchy Visualisation

Computer-based visualisation is an interdisciplinary research area involving computer science, graphics theory, data analytics, visual design, cognitive science and human-computer interaction domains (Ware, 2012). It uses different visual marks such as points, lines and areas, and visual channels like position, colour, shape and size, to encode data, combined with different alignments, to create effective visual representations to support user tasks (Munzner, 2014). Effectiveness is a goal that visualisations should aim for, which emphasises the accuracy, completeness and efficiency of task performance. Appropriate combinations of visual marks, channels and alignments affect the effectiveness of visualisations. Space efficiency, which is an important consideration related to the scalability of a visualisation, is also associated with these combinations. The discussion of visualisations in this section will focus on these two measures.

For hierarchical structures, the common visual marks are *node* and *link* (e.g., layered tree in Table 3.1), or *area* (e.g., traditional treemap in Table 3.1), and the common visual channels are *connection*, *containment*, *adjacency* and *hybrid of the above* (Munzner, 2014; McGuffin and Robert, 2010; Schulz et al., 2011). Schulz et al. (2011) defined the alignments of visual marks for hierarchy visualisations as *layered*, *radial* and *free*. Table 3.1 represents a matrix containing some examples for different combinations of visual marks, channels and alignments. The following discussions are organised based on the visual channels. The described visualisations not shown in dedicated figures are illustrated in Table 3.1.

#### Connection

Visualisation with node-link connection is one of the most well-known hierarchy representations, in which nodes represent classes and edges represent relationships. With

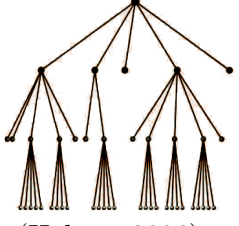
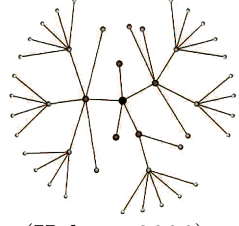
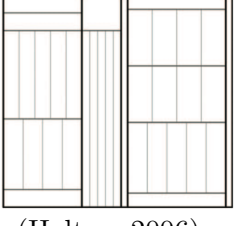
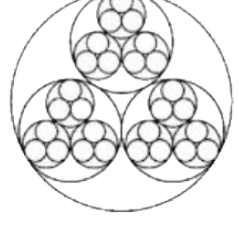
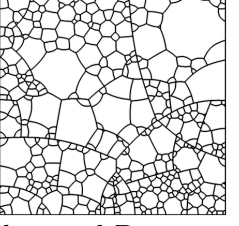
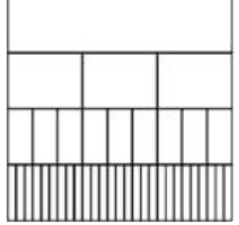
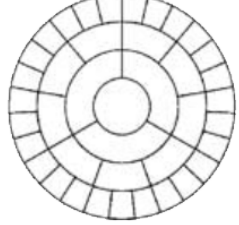
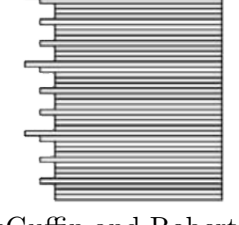
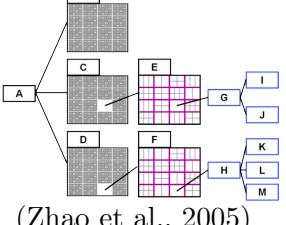
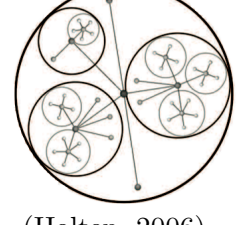
Alignment Channel	Layered	Radial	Free
Connection	 <p>(Holten, 2006) name: layered tree marks: node, link</p>	 <p>(Holten, 2006) name: radial tree marks: node, link</p>	
Containment	 <p>(Holten, 2006) name: traditional treemap marks: area</p>	 <p>(McGuffin and Robert, 2010) name: bubble tree marks: area</p>	 <p>(Balzer and Deussen, 2005) name: Voronoi treemap marks: area</p>
Adjacency	 <p>(McGuffin and Robert, 2010) name: icicle plots marks: area</p>	 <p>(McGuffin and Robert, 2010) name: concentric plots marks: area</p>	
	 <p>(McGuffin and Robert, 2010) name: indented list marks: area</p>		
Hybrid	 <p>(Zhao et al., 2005) hybrid of connection and containment name: elastic tree marks: node, link, area</p>	 <p>(Holten, 2006) hybrid of connection and containment name: balloon tree marks: node, link, area</p>	

Table 3.1: Examples of hierarchical structure representations

the layered alignment (layered trees), lower-level nodes are placed below their related higher-level nodes (Reingold and Tilford, 1981), while with radial alignment (radial trees), nodes are positioned on concentric circles based on the degree of their levels (Herman et al., 2000). Burch et al. (2011) conducted a user study to investigate the effectiveness of layered trees and radial trees for solving a hierarchy exploration task of finding the lowest common ancestor of a given set of nodes. An eye tracking device was employed to identify user behaviours. The results showed that layered trees significantly outperformed radial trees in respect of both accuracy and completion time. The eye tracking data revealed the participants tended to cross-check their answers more frequently with radial trees than with layered trees, which was a sign of difficulties with interpreting a radial tree, whose hierarchical structure was not as clear as in layered depictions and was also a less common representation.

### **Containment**

Although radial trees employ the available space more efficiently than layered trees, node-link connection does not make an optimal use of the visual space. Traditional treemaps (Shneiderman, 1992), bubble trees (Wang et al., 2006), and Voronoi treemaps (Balzer and Deussen, 2005) place all the children of a node enclosed within the area assigned to that node, so as to display the hierarchical structure by means of containment rather than connection, which makes a somewhat more efficient use of the available space. Traditional treemaps maximise the space efficiency by utilising a space-filling technique, which uses the whole screen space and subdivides the space of a node for its children.

However, using traditional treemaps to display the hierarchy makes it more difficult for users to perform tasks focusing on the topological structure like tracking the path through the tree. To solve this important limitation, nested treemaps (Demian and Fruchter, 2006), which give margins surrounding child nodes in treemaps (Figure 3.1 (left)), and cascaded treemaps (Lü and Fogarty, 2008), which place cascaded rectangles by leaving a small boundary between a node and its children (Figure 3.1 (right)), have been developed to put more emphasis on the hierarchical structure. With this trade-off between space efficiency and tree structure perception, nested treemaps and cascaded treemaps do not work as well with deep hierarchies.

Neumann et al. (2005) proposed ArcTrees that present the hierarchy structure using a one dimensional treemap (see Figure 3.2). This one-dimensional treemap organises the child nodes horizontally within their parent nodes and some vertical offset is given between the parent and child nodes. This representation can show hierarchical structure more clearly than two-dimensional treemaps and save some vertical space. However, it faces the scalability problem for visualising large hierarchies and particularly wide hierarchies (containing many leaves).

### **Adjacency**

With area adjacency, child nodes are drawn adjoining with their parent nodes. Icicle plots (Kruskal and Landwehr, 1983) visualise the hierarchy structure by horizontally aligning child nodes under their parents, such that the areas of child nodes are partitions

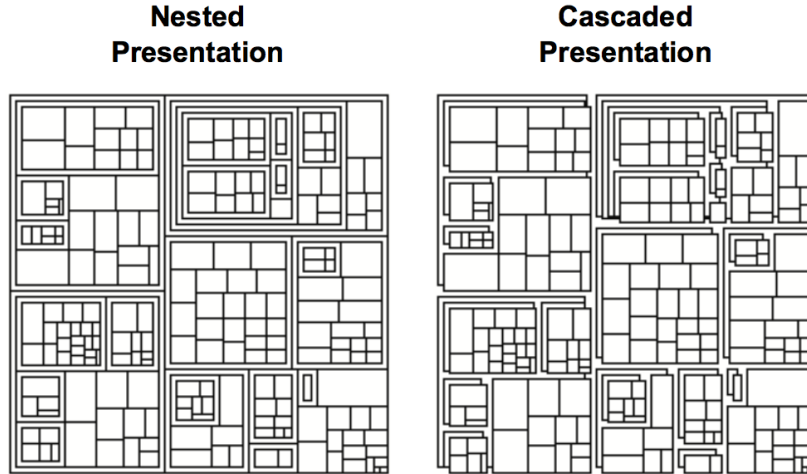


Figure 3.1: Nested treemap and cascaded treemap (Lü and Fogarty, 2008).

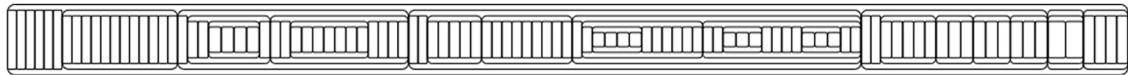


Figure 3.2: An example of the one dimensional treemap representation in ArcTrees (Neumann et al., 2005).

of the areas of their parent nodes. As the available width of the visual space for the root limits the space for the entire plot, when the hierarchy grows wider and there are too many leaves, those leaves may be squeezed together. This problem is somewhat solved with concentric plots (Stasko and Zhang, 2000) by adopting a radial variation with circle sections, where the available space for leaves partitions the circumference.

Similar to icicle plots, indented lists (Smith et al., 1984) show the hierarchy structure with relative vertical positions. Nodes are rectangles with constant height, stacked in the top-bottom order, based on a depth-first traversal of the hierarchy. Each rectangle is indented horizontally to the right corresponding to its depth. While the number of nodes increases, such a representation will become very tall. Indented list representations are popular in file system browsers to show directory hierarchies.

McGuffin and Robert (2010) conducted a systematic comparison of space efficiency for the representations mentioned above. In their findings, when using total area as the metric, which calculated the percentage of the utilised area in the total area, traditional treemaps, nested treemaps and then icicle plots used the screen space most efficiently. Indented lists were ranked the fourth in efficiency of total area, followed by bubble trees and concentric plots. Layered trees and radial trees were ranked as the least efficient and the second least efficient, respectively. This ranking changed when using another metric, leaf node area, which defined the minimum area of the leaf nodes. Traditional treemaps remained the first place in the ranking with this metric, while layered trees and radial trees remained the bottom two places. Indented lists were ranked second. The ranking for nested treemaps and bubble trees was lower for the obvious reason that they used additional space on the margin between nodes. Icicle plots were shown to be less efficient



in terms of leaf node area than concentric plots, contrary to the result when comparing their total areas.

### Hybrid

Hybrid visualisations combine the best features of different representations for different parts of data based on the data structure and features, allowing users to view each part of the data in the most effective way.

For example, layered trees can clearly and intuitively show the topology of a hierarchy, but do not use space efficiently. They leave white space between upper level nodes and pack lower level nodes densely, so fail to scale for large hierarchies. In contrast, treemaps are the most space-efficient but they are hard to interpret and distinguish the different levels of a hierarchy. Elastic tree (Zhao et al., 2005) combines these two representations to offer a trade-off between an intuitive display and efficient space usage for visualising hierarchies. It uses node-link connection to visualise the higher-level nodes as many as possible. When the hierarchy becomes too dense in the deeper subtrees, treemaps are adopted to represent the lower levels. It also allows users to select and emphasise interesting structures and content, displaying in a flexible and user preferred manner, at the cost of dual visual representations.

Similarly, balloon trees (Lin and Yen, 2007) hybridise radial trees and bubble trees, to solve the low space-efficient problem of bubble trees. Balloon trees represent the leaf nodes with node-link marks and connections instead of circle area marks and containment, to reduce the occupied space of leaves. With the node-link connection form, leaves under the same parent are placed on the circumference of an invisible circle centred at their parent. Since balloon trees are parent-centrally nested, it is difficult to perceive the hierarchical structure. Also, the nodes on deep levels can be difficult to see since they are given very little space.

## 3.2 Ontology Visualisation

Compared to visualising strict hierarchies, visualising ontologies is not easy. Firstly, the multiple inheritance in ontologies poses a difficult problem that rules out a whole class of visualisation techniques. Multiple inheritance is often solved by duplicating a class under each of its parents or using multiple edges to link a class to all of its parents. Both have drawbacks. With duplication, redundancy occurs; while with multiple edges, there will be increased visual clutter. Furthermore, as discussed in Chapter 2, besides hierarchical inheritance relationships, ontologies can contain homogeneous and heterogeneous associations, and also be enriched with other relationships such as equivalence and disjointness. In addition, in some ontologies, the ontology classes may have up to thousands of instances (Katifori et al., 2007). Depending on the task, sometimes all these instances need to be visualised.

A comprehensive survey on ontology visualisation (Katifori et al., 2007) categorises systems for visualising ontologies based on their visualisation types: indented list, node-link and tree, zoomable, space-filling, focus + context or distortion, and 3D information

landscapes. A recent survey (Saghafi, 2016) proposes two categories: graph-based methods and multi-method visualisation techniques. The latest survey (Dudáš et al., 2018) provides a useful classification and comprehensive evaluation of available ontology visualisation tools. The results show that most visualisation systems focus on class hierarchies. Some of them visualise all the relationships, while some exclusively visualise the hierarchy. One major challenge is the scalability of the tools is still limited. Also, their usability and the tasks and use cases they can support are often unclear.

This thesis groups and discusses ontology visualisation tools based on different visual channels mentioned above: connection, containment, adjacency and multiple views that employ different channels in separate views. Although this research addresses the visualisation of single ontologies and their associations, prior work focusing on the visualisation of multiple ontologies is included in this review. In the literature of multiple ontology visualisations, the discussions mainly address the approaches for presenting links between ontologies, while the discussions in this thesis will centre on the representations of individual ontologies in these works. Many ontology visualisations have been developed. This review only chooses some representative examples to discuss, as the others are similar to the ones that are reviewed.

A few tools have been developed to model the relations in ontologies via visual notations. These kinds of visual notations normally model all the axioms in ontologies, such as inheritance hierarchical relationships, property value restrictions, property cardinality restrictions, literal axioms, equivalence and disjointness axioms, and do not differentiate them using distinct representations. Figure 3.3 and 3.4 show two noteworthy examples. As this research focuses on ontology visualisations, these visual notations that are more related to visual languages will not be discussed in detail.

Some visualisation systems do differentiate ontology associations from the ontology hierarchy structure, by using different visual marks, channels, colours or even dimensions. The discussion of these systems is in the following sections.

### 3.2.1 Connection

Key Concept Visualisation (KC-Viz) (Motta et al., 2012) is an ontology visualisation tool based on node-link connection. KC-Viz places the node with the highest information richness, which is measured by the density of relations of a node, at the centre of the visualisation. Then it arranges other nodes around the centre from middle out (see Figure 3.5). The authors evaluated KC-Viz with a user study (Motta et al., 2011). 21 participants were asked to perform tasks that were designed to cover different exploration strategies, using KC-Viz and OWLViz. OWLViz (Horridge, 2005) is the default visualisation in Protégé, which uses node-link connection with layered alignment to show the ontology hierarchy (see Figure 3.6). An ontology, which included 630 classes, was used as the data. The results show the task completion time for KC-Viz is shorter on average, and the satisfaction level of KC-Viz is higher than OWLViz. Although the results favour KC-Viz, it still suffers the clutter issue with large ontologies, which is intrinsic to approaches based

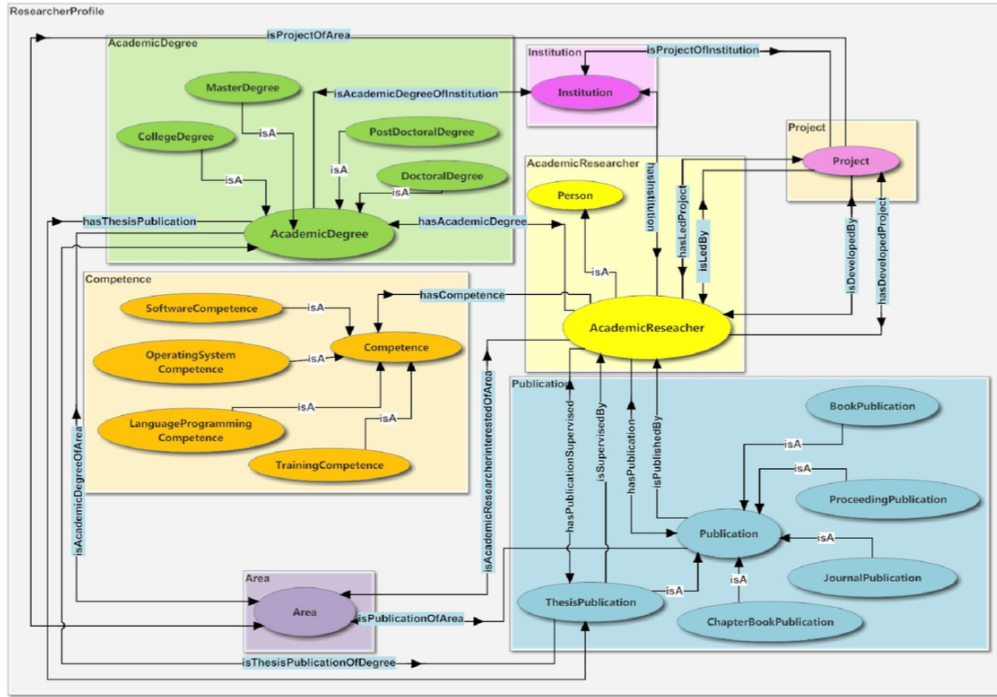


Figure 3.3: Onto Design Graphics (ODG) (Silva-López et al., 2014) based on Unified Modeling Language (UML) component.

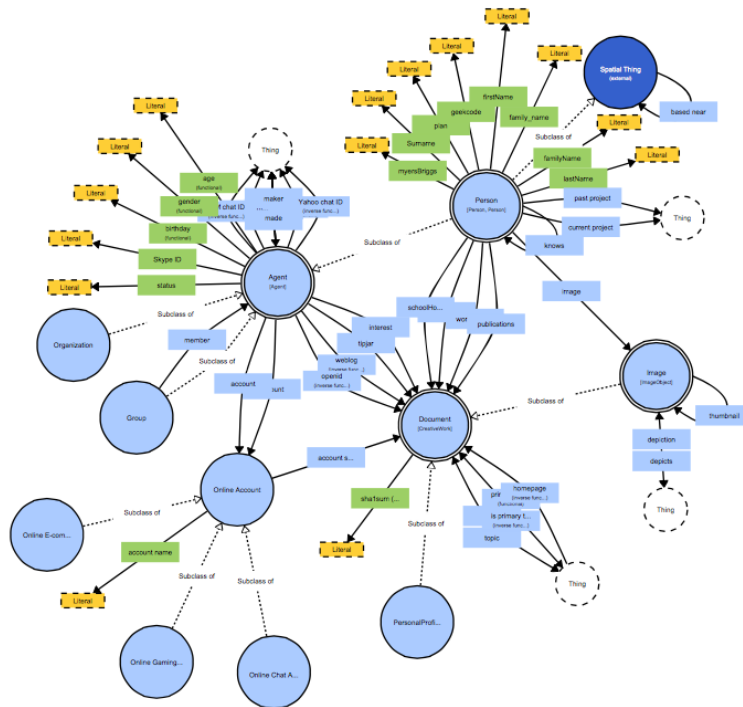


Figure 3.4: Visual Web Ontology Language (VOWL) (Lohmann, Negru, Haag and Ertl, 2014) based on a well-defined user-friendly set of graphical primitives and colour scheme.

on node-link connection. Also, as the nodes are positioned based on information richness, the hierarchical structure of the ontology is not clearly shown.



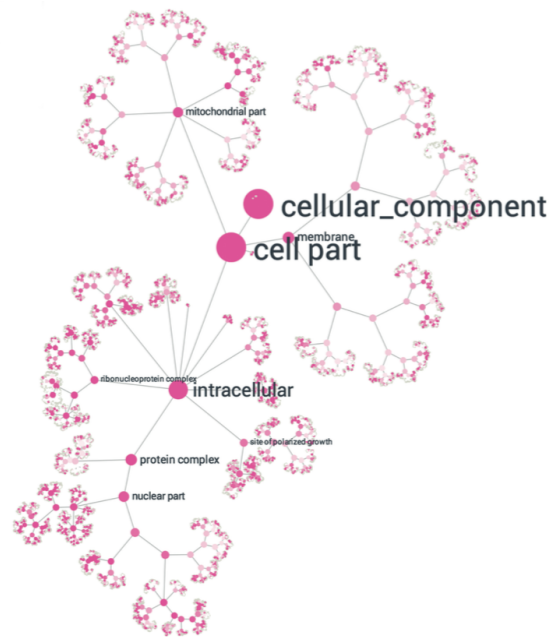


Figure 3.7: NeXO Web visualisation of Gene Ontology (Dutkowski et al., 2013)



Figure 3.8: Treemap of GO (Babaria, 2004). Size shows the amount of fold change for genes and colour shows gene acidity: green (basic) or red (acidic).

CropCircles are represented as circles. The circles of lower level nodes are nested inside the circle of their parents with smaller size. The space given to each node is based on the size of its subtrees. Then the child circles are ordered from the largest to the smallest inside the parent node (see Figure 3.9). CropCircles provides a good overview of an ontology and its topology at once. It is also zoomable, by employing a selection pane that let users drill down the hierarchy level by level. Users can also specify which top levels

are shown in the visualisation. The authors conducted an empirical evaluation with 18 subjects. The chosen data were two ontologies, with around 2000 classes, a maximum depth of 11 and significantly different topologies. The experiment found out CropCircles outperformed nested treemaps in topological tasks. In addition, it performed well against nested treemaps in tasks requiring participants to find the deepest node, because it shows the topology more clearly.

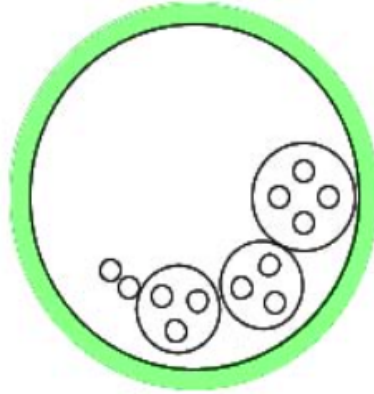


Figure 3.9: CropCircles visualisation (Wang and Parsia, 2006)

### 3.2.3 Adjacency

The most common use of adjacency for visualising ontologies are indented lists. With an indented list, classes are presented as nodes. The whole list is expandable and retractable, which is similar to the file browser in Windows Explorer. Since it benefits from familiarity, it is not surprising that an indented list is often provided as a complementary visualisation alongside the main visualisation in the majority of ontology visualisation systems. Figure 3.10 shows an example of an indented list in Protégé (Noy et al., 2000) for visualising the classes in the Pizza ontology.

Jia et al. (2010) developed an Enhanced Radial Space-Filling (ERSF) layout to visualise biological ontologies. They employed a 3D radial space-filling approach that is similar to concentric plots discussed in Section 3.1, where the height of the nodes presents attribute values. ERSF uses an orbit metaphor to visualise multiple inheritance (see Figure 3.11). The parent adjacent to a class is its primary parent and the other parents linked by the orbit pathways are secondary parents. Extended green edges are drawn from the centre of the secondary parents to the orbit of the child class, connected by red points. To distinguish the orbits, they are coloured the same as the corresponding child classes, as a sort of colour wheel.

### 3.2.4 Multiple Views

In ontology visualisations, the multiple views approach, which employs different representations in separate views, normally serves one of two purposes: to visualise different parts



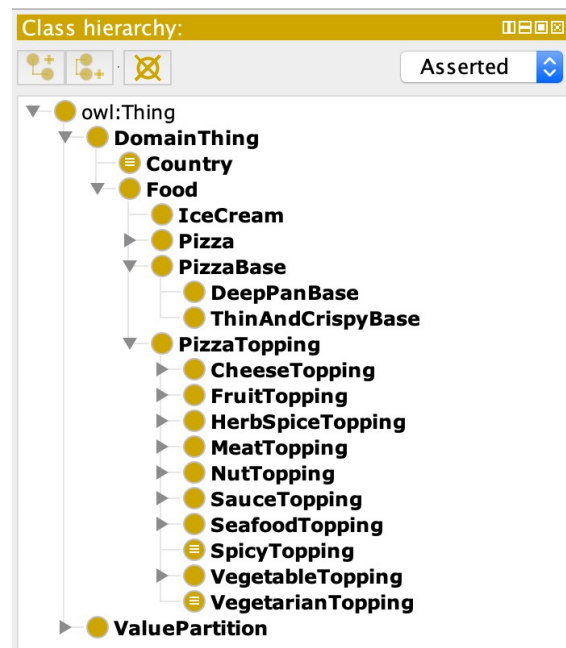


Figure 3.10: Indented list in Protégé, showing the Pizza ontology.

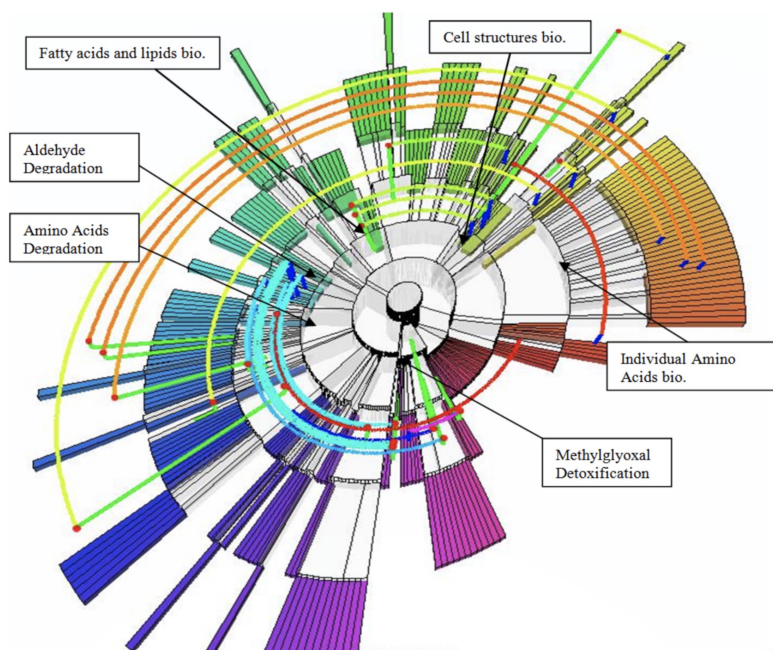


Figure 3.11: ERSF visualisation (Jia et al., 2010).

of an ontology in different views, or to facilitate users understanding about the same part of an ontology using different techniques in different views.

An example of multiple views is Jambalaya (Storey et al., 2001), which is a visualisation plug-in for Protégé. In its main view with SHriMP (Simple Hierarchical Multi-Perspective) (Storey et al., 1997), the inheritance hierarchical structure of classes and instances is represented as a set of nested rectangles, and the associations between classes and instances are drawn as edges between them (see Figure 3.12a). Different colours are encoded to distinguish classes (blue) and instances (pale yellow). The colours are also used to represent

different types of associations. One downside is that this nested rectangle view cannot show deep level nodes clearly without any user interactions. To provide an alternative overview of the hierarchy structure, Jambalaya uses another view with node-link connection, where the hierarchical relations are aligned as layers and the associations are still drawn as edges on top of the hierarchy (see Figure 3.12b). Jambalaya allows users to filter classes, instances and relations for display. In a later evaluation conducted by Akrivi et al. (2006), comparing three Protégé visualisation plug-ins Jambalaya, TGVizTab (node-link connection) (Alani, 2003), OntoViz (node-link connection) (Sintek, 2003), and the default indented list view, Jambalaya got positive results. In this evaluation, participants were asked to perform a set of tasks related to ontology hierarchy, classes, instances and associations. Jambalaya performed similarly to TGVizTab and better than OntoViz on the property related tasks. However, participants commented that they disliked the appearance of edges, and these links started to obscure information in the case of many classes and instances.

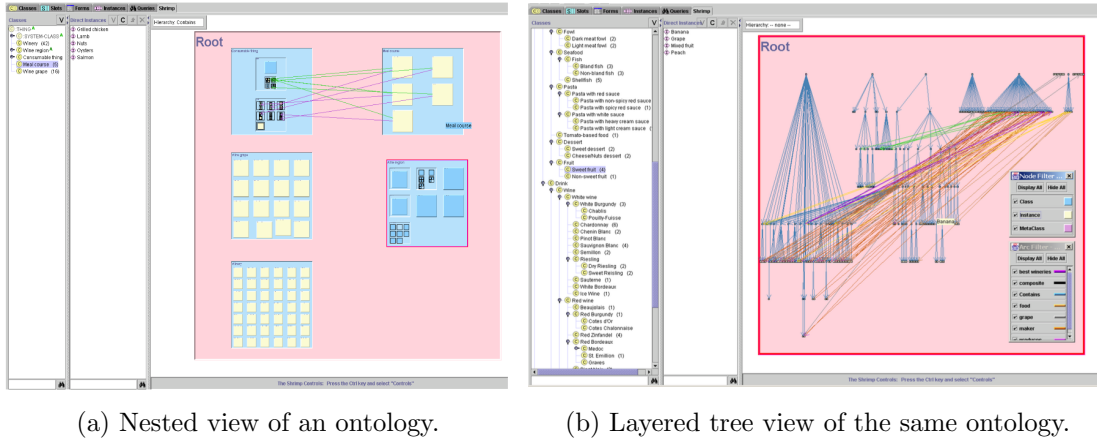


Figure 3.12: Jambalaya interface in Protégé (Storey et al., 2001)

A similar approach to the SHriMP view is identified in a later tool, Knoocks, proposed by Jurčík (2012), where class hierarchy is visualised as a set of icicle plots and the associations are drawn as edges between them (see Figure 3.13). These icicle plots are oriented horizontally, where the child classes are adjacent to their parent class on the right side. They can be folded to save space and expanded to display class labels. Knoocks places the icicle plots on a circular layout and draws the edges as curves, which helps reduce the overlaps between classes and edges.

Another example is AIViz (Lanzenberger et al., 2010) plug-in for Protégé, visualising mappings between two ontologies. It has four separate views: two indented lists for the class hierarchies (see Figure 3.14 (a)) and two node-link representations visualising the associations between classes (see Figure 3.14 (b)). The node-link representation uses a force-directed layout to place the nodes, which aims to minimise the length of edges and edge crossings (Fruchterman and Reingold, 1991). In the node-link representation, the nodes are clustered classes based on the selected level of clustering adjusted by the sliders on the right, and the edges represent the selected association. The size of the nodes indicates the number of classes in them, and the colour encoding represents the type of



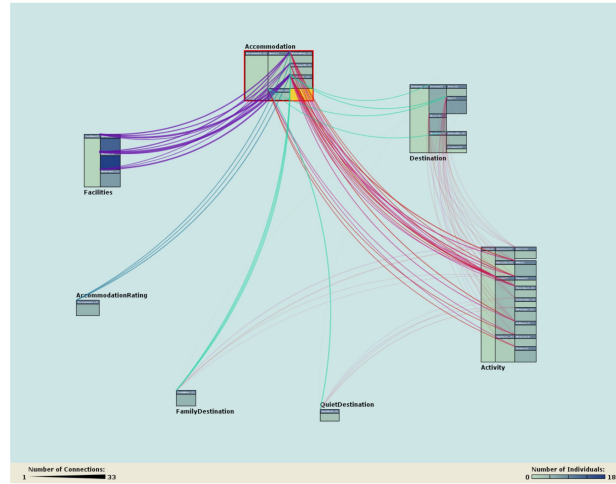


Figure 3.13: Knoocks visualising an ontology hierarchy and associations (Jurcik, 2012).

mappings between the two ontologies, as discussed in Section 7.3.6. These four views are connected by linking and brushing (Becker and Cleveland, 1987; Buja et al., 1991; Wybrow et al., 2014), where selecting a node in one view results in highlighting the corresponding nodes in the other three views. AIViz can only visualise one type of associations at a time per user selection.

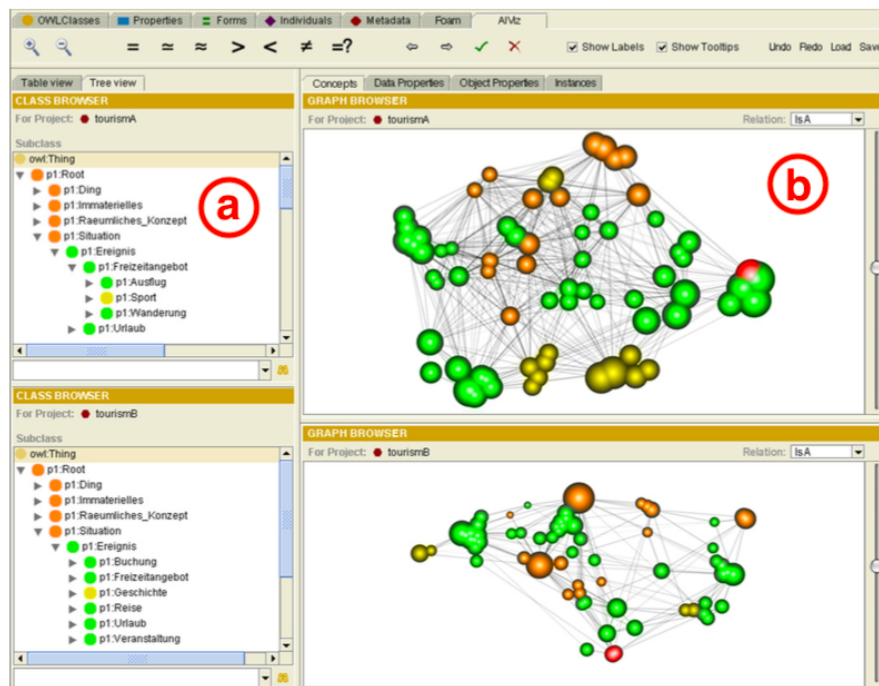


Figure 3.14: AIViz interface (Lanzenberger et al., 2010)

Similarly, OntoViewer (da Silva et al., 2012) uses an indented list (see Figure 3.15 ①) and a 2D radial node-link tree (see Figure 3.15 ②) to visualise the class hierarchy, and a 2.5D radial tree for the associations (see Figure 3.15 ③). The layout of the classes in the 2.5D radial tree is the same as in the 2D radial tree, based on the ontology hierarchy structure. The associations are drawn as curves in the third dimension. The interface

includes a list of properties, allowing users to select the types of associations to visualise. If multiple properties are selected, different colours are used to distinguish the types of associations in the 2.5D radial tree. Although this 2.5D radial tree allows users to rotate the view, the association links are still not easy to follow.

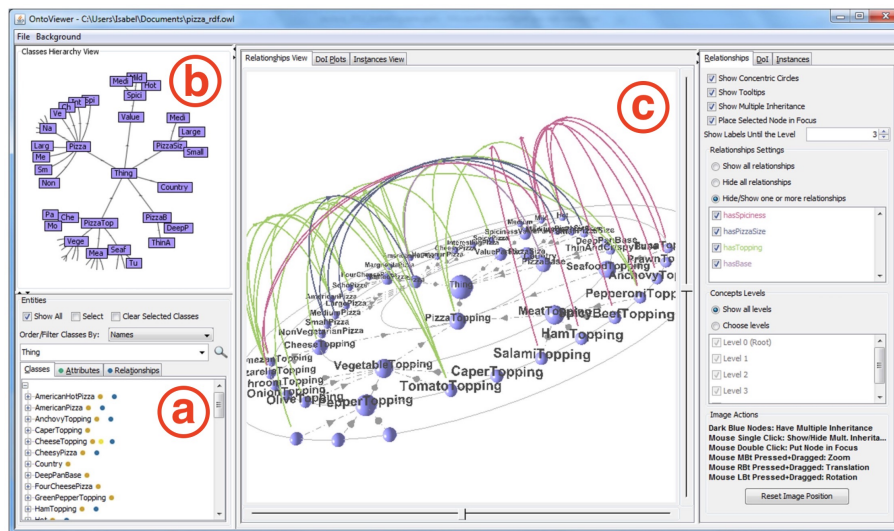


Figure 3.15: OntoViewer interface (da Silva et al., 2012)

Kuhar and Podgorelec (2012) proposed a three-view visualisation consisting of hierarchical circle (see Figure 3.16 (a)), indented list (see Figure 3.16 (b)) and node-link diagram (see Figure 3.16 (c)). In contrast to concentric plots discussed in Section 3.1, in the hierarchical circle approach, child classes are drawn as inner circles adjacent to the outer circle representing their parent class. Associations are presented as edges connecting the inner circles. It is unclear how to draw associations between intermediate classes with this hierarchical circle. Also, the scalability of it is limited, as inner circles will become smaller when the depth of ontology hierarchy increases, so less space will be available for deep nodes. The view can be switched between T-Box mode and A-Box mode to show relationships between either classes or instances. In the indented list approach, only the ontology hierarchy is shown by default. When users select a class, the associations applied to this class will be drawn. The node-link diagram visualises both hierarchical relationships and associations. The classes are positioned in a way to minimise the length of links and the number of line crossings. Functionality for filtering the number of hierarchy levels, instances and properties, zooming, and collapsing or showing all, are provided. Users can switch the main view between the three approaches. Animation is employed to help users comprehend the transition when they change the views.

Another tool was developed by Jiao et al. (2013) as a plug-in to the Dunnart constraint-based diagram editor (Dwyer et al., 2008). This tool visualises large ontologies via three views: landmark view, local view and axiom view. The landmark view utilises an orthogonal layered tree to visualise a simplified sub-ontology that only contains landmark nodes, namely the key concepts. These concepts are extracted from the original ontology based on importance scores that are calculated by a set of static and dynamic measures. In order to further save space, if the number of leaves for a single parent node is above

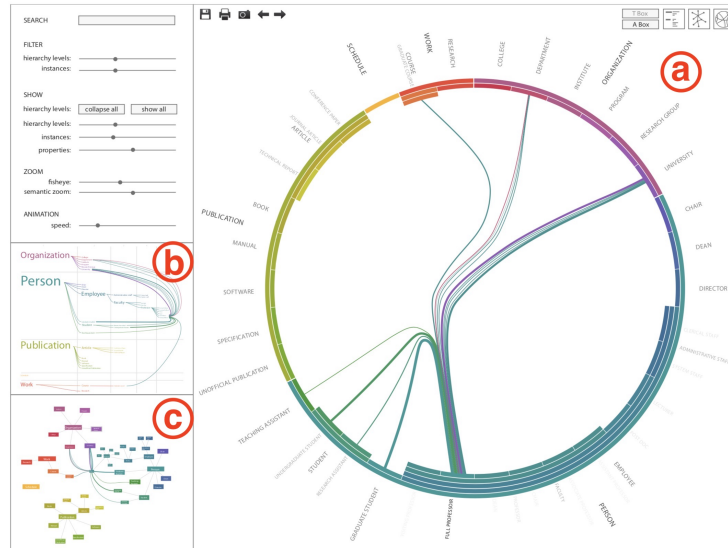
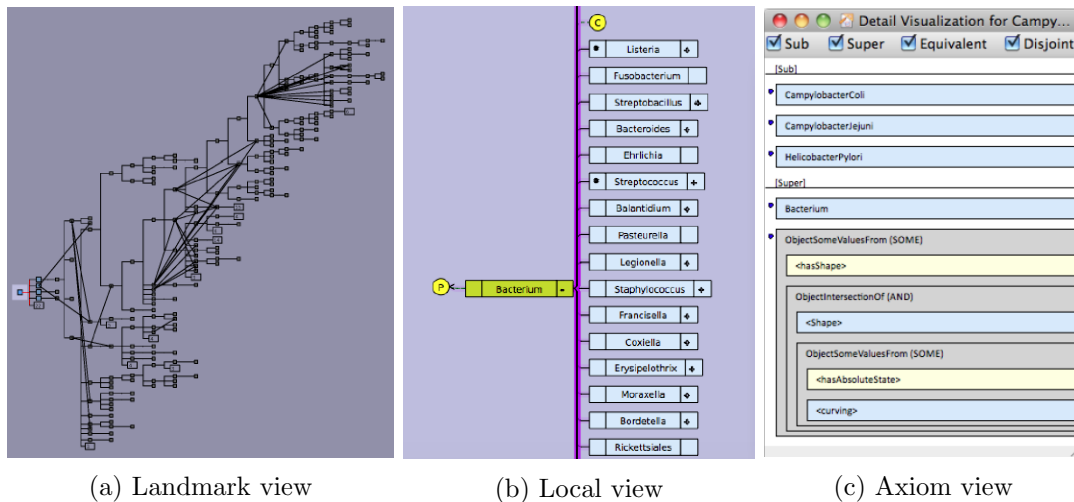


Figure 3.16: The three-view visualisation proposed by Kuhar and Podgorelec (2012), visualising an ontology having 45 classes with a class selected.

a certain threshold, these leaves will be collapsed to a single large node attached to their parent. The multiple inheritance in the landmark view is represented as non-orthogonal edges above the main hierarchy orthogonal edges, directly connecting the child node with its other parents (see Figure 3.17a). The local view visualises a subset of the ontology that a user is interested in, using a layered tree, with nodes aligned by the constrained network layout (Dwyer et al., 2006) and edges drawn by orthogonal edge routing (Wybrow et al., 2009) to avoid overlaps between nodes and edges (see Figure 3.17b). The axiom view shows the syntax of the important axioms that are related to the focal concept (see Figure 3.17c). This tool provides flexible navigation between the three views and records the navigation history. It effectively visualises the overview structure as well as the detailed focal points, and avoids clutter for large ontologies. However, since the ontology hierarchy is trimmed a lot in the landmark view and there is no information provided, the size of the hierarchy structure is unclear in the visualisation.



(a) Landmark view

(b) Local view

(c) Axiom view

Figure 3.17: Ontology visualisation in Dunnart (Jiao et al., 2013)

OntoTrix (Bach et al., 2011) is designed to visualise associations but on the instance level. It has four views: NodeTrix view (see Figure 3.18 (a)), minimap view of NodeTrix (see Figure 3.18 (b)), class hierarchy view (see Figure 3.18 (c)) and property hierarchy view (see Figure 3.18 (d)). NodeTrix is a hybrid network visualisation introduced by Henry et al. (2007), which uses matrices for dense subgraphs and node-link representation to link these subgraphs. OntoTrix employs this hybrid visualisation to show the associations between instances. Different colours are used for different association types. In the matrices, the intersection cell is coloured if two instances have an association. If two instances are linked by more than one type of association, the cell will be sliced horizontally, with different slices coloured differently. As the NodeTrix view can be zoomed, the minimap view gives an overview of the whole NodeTrix and preserves the context for the elements in the NodeTrix view. The class hierarchy is visualised by a node-link diagram view. As ontology properties can also be constructed in terms of the inheritance hierarchy, a node-link radial tree is used to visualise the property hierarchy. All these four views are coordinated through linking and brushing. OntoTrix effectively addresses the scalability issue for visualising large instance sets contained in ontologies, but the node-link diagram view fails to deal with large ontology hierarchies.

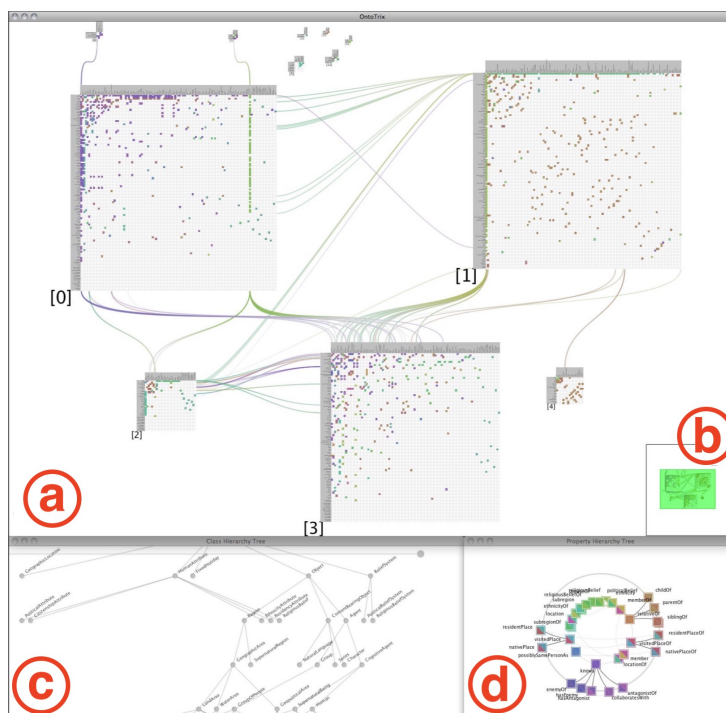


Figure 3.18: OntoTrix interface (Bach et al., 2011).

### 3.2.5 Summary

Table 3.2 summarises the characteristics of the tools for visualising ontologies and associations discussed above.

Tool Name	Visual Marks	Visual Channel	Alignment	Association	Multiple Inheritance	Instance	Reference
KC-Viz	node, link	connection	middle-out	heterogeneous associations	multiple edges	not supported	Motta et al. (2012)
OWLViz	node, link	connection	layered	not supported	multiple edges	not supported	Horridge (2005)
NeXO Web	node, link	connection	radial Pythagoras tree drawing	not supported	duplication	not supported	Dutkowski et al. (2013)
GO treemap	rectangle	containment	layered	not supported	not supported	supported	Babaria (2004)
CropCircles	circle	containment	radial ordering	not supported	duplication	not supported	Wang and Parsia (2006)
Protégé indented list	rectangle	adjacency	layered	not supported	duplication	not supported	Noy et al. (2000)
ERSF	arc, link	adjacency, connection	radial	not supported	multiple edges	not supported	Jia et al. (2010)
Jambalaya	rectangle, node, link	containment, connection	layered	heterogeneous associations	duplication	supported	Storey et al. (2001)
Knoocks	rectangle, link	adjacency, connection	layered, free	heterogeneous associations	duplication	supported	Jurcik (2012)
AlViz	rectangle, node, link	adjacency, connection	layered, force- directed	homogeneous associations	duplication	supported	Lanzenberger et al. (2010)
OntoViewer	rectangle, node, link	adjacency, connection	layered, free	heterogeneous associations	duplication	supported	da Silva et al. (2012)
Three-view tool	arc, node, link	adjacency, connection	radial, layered, free	heterogeneous associations	duplication	supported	Kuhar and Podgorelec (2012)
Landmark	node, link, syntax	connection	constraint lay- ered	heterogeneous associations	multiple edges	not supported	Jiao et al. (2013)
OntoTrix	node, link	connection, hybrid	layered, free	heterogeneous associations	unknown	supported	Bach et al. (2011)

Table 3.2: Summary of ontology visualisation tools

### 3.3 Visual Compression

When dealing with large ontologies, several approaches can be used to address the visual clutter problem. These approaches were initially developed for large graphs. They can be categorised into two main strategies: manipulate view and visual compression.

Manipulate view (Munzner, 2014) often operates on the view level, either reducing data items for display or navigating to a different viewpoint. With item reduction, less important data is removed from the visualisation. This can also be done by users using filtering. Navigation changes the location of the viewpoint, often involving user interactions such as zooming, panning, or searching.

Visual compression changes the data structure for visualisation, which is helpful for reducing visualisation complexity and finding meaningful patterns. There are three major approaches for visual compression: aggregation, clustering and simplification. Shneiderman and Dunne (2012) defined *aggregation* as organising data items into groups based on data attributes and/or replacing the groups with single elements; *clustering* as organising data items into clusters based on structural connectivity algorithms and/or replacing the clusters with single elements; and *simplification* as replacing common structural patterns with simplified glyphs.

#### 3.3.1 Aggregation

Attribute-based aggregation groups data items to give insights into data attributes and reveal patterns. This approach is normally used for data with properties. Linage (Nobre et al., 2018) shows an example of using aggregation for hierarchically structured multivariate data. The hierarchy is visualised as a node-link layered tree on the left, orientated horizontally (see Figure 3.19). If a node has certain attributes, this node is considered *interesting*. The interesting nodes are given distinct rows, while other nodes are grouped together into one row using a short line to separate parent and child nodes. A recursive algorithm runs through the hierarchy to identify different subtree structures around interesting nodes, for example, a leaf node is interesting (see Figure 3.19 node ⑥) and an intermediate node is interesting (see Figure 3.19 node ③). Simply speaking, if a node is interesting, the branch containing it will be split from its siblings. The attributes for a node are presented as blocks on the row of the node, with colours indicating attribute values.

#### 3.3.2 Clustering

Clustering by structural connectivity is a widely used approach for graph visualisations. With clustering, each produced cluster is a subset of nodes, which is normally replaced by a metanode. The edges between metanodes are produced by combining edges to nodes within the metanode. Archambault et al. (2010) visualised a network (see Figure 3.20a) using Path-Preserving Clustering (PPC) algorithm (Archambault et al., 2008), shown in Figure 3.20b. The size of metanodes indicates the number of contained nodes, and the thickness of edges indicates the number of connections. The colours encode attribute values for both nodes and metanodes. Users can “open” metanodes to investigate their contents.

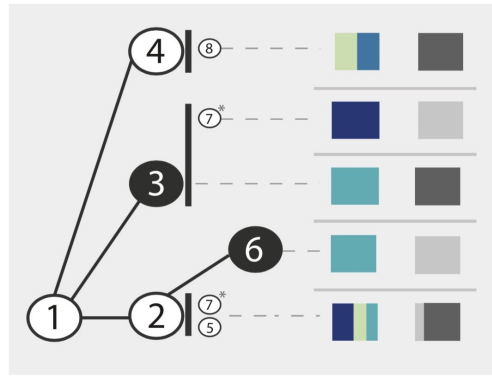


Figure 3.19: An example of Lineage aggregation (Nobre et al., 2018). The interesting nodes are coloured black.

The authors conducted a user study to evaluate how this clustering approach affected the visualisation readability. The participants were asked to perform some tasks related to reading global and local structures with attributes. The results showed that there was no significant difference when the clustering was used. However, if the network is highly connected, in other words, there are many connections between nodes, the clustering can improve performance, as the visualisation complexity is significantly reduced.

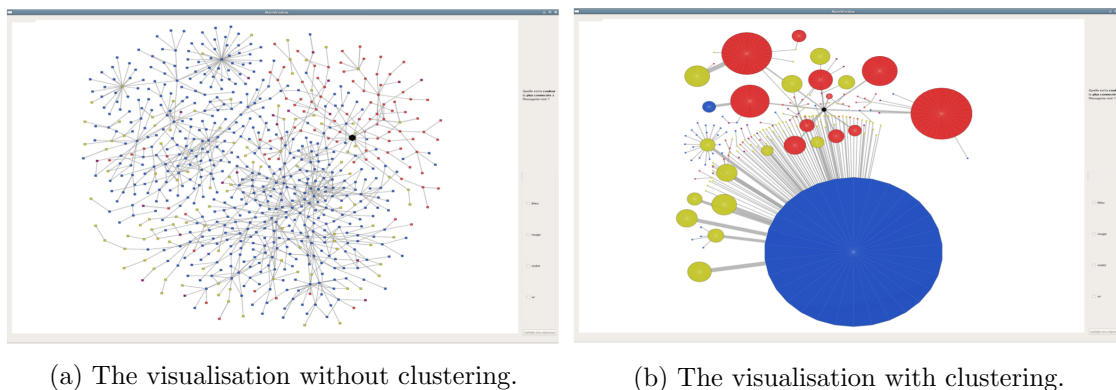


Figure 3.20: Clustering of a network (Archambault et al., 2010).

### 3.3.3 Simplification

Simplification refers to the practice of replacing common patterns of nodes and edges with compact and meaningful glyphs. A common pattern is known as a motif (Dunne and Shneiderman, 2013), and a glyph is defined by Munzner (2014) as an object made of multiple visual marks. Simplification with glyphs raises the visibility of common structures, and also reduces visualisation complexity.

Fuchs et al. (2014) defined three variations of a glyph: contour only, structure only and contour + structure. Figure 3.21 shows an example of these three variations for a star glyph. The authors conducted user studies to investigate the effect of using contour on similarity perception of star glyphs. The results showed that glyphs without contours



made the tasks of detecting data similarity easier, while glyphs with contours facilitated the tasks of finding shape similarity.

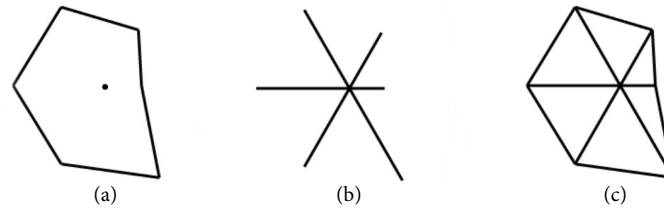


Figure 3.21: Glyph variations (Fuchs et al., 2014): (a) contour only, (b) structure only, (c) contour + structure.

Dunne and Shneiderman (2013) developed a tool to automatically identify common motifs in networks: fan motif, connector motif and clique motif, and simplify them into glyphs with different shapes (see Figure 3.22). A fan motif consists of a head node and its leaf nodes. Replacing fan motifs with fan glyphs can reduce the visualisation complexity dramatically if there are hundreds or thousands of leaves. A connector motif consists of anchor nodes and their span nodes. The replacement of connector motifs with connector glyphs can simplify a network with a dense centre. Clique glyphs are used to replace clique motifs, which is a complete graph where each pair of nodes is connected by at least one edge. Replacing clique motifs with clique glyphs can make the overall connectivity of a network easier to understand.

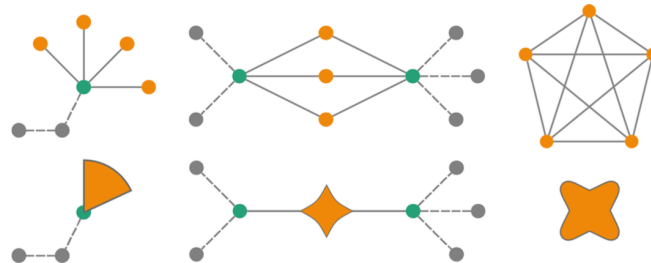


Figure 3.22: Fan, connector and clique motif (top) and their simplification glyphs (bottom) (Dunne and Shneiderman, 2013).

An example of motif simplification for visualising a network is shown in Figure 3.23. Figure 3.23 (a) shows the initial network, and (b) shows a simplified version. The simplified network has 25 nodes instead of 513 nodes in the original network and requires less screen space. Users can click to investigate the contents of any glyphs and align the glyphs manually. A user study conducted with this representation with a set of tasks related to topology, attributes and overview, gave a positive result. Finding nodes by their labels and finding the shortest path length was significantly more accurate and faster with the motif simplification approach than without it. Participants estimated the number of nodes in the simplified network more accurately but more slowly than in the original network.

One tool, which applies motif simplification to visualise hierarchies, called SpaceTree, was developed by Plaisant et al. (2002). SpaceTree visualises hierarchies with layered trees. Subtrees that are too large to display in the available screen space are replaced by



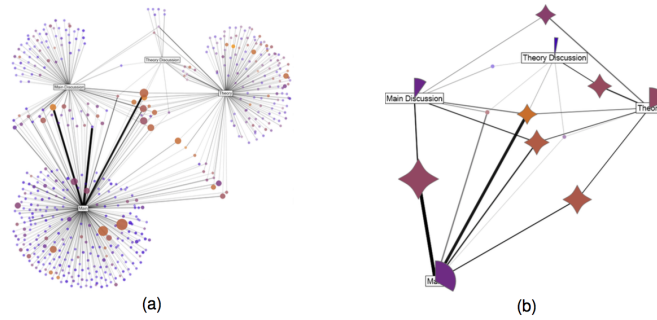


Figure 3.23: Motif simplification (Dunne and Shneiderman, 2013): (a) the original network (b) a simplified version using fan and connector glyphs.

triangular glyphs (see Figure 3.24). Darker glyphs indicate the subtrees with more nodes. Taller glyphs depict deeper subtrees, while wider glyphs depict subtrees with more leaves. When clicking a glyph, the hidden subtree is expanded with the maximum number of levels that can be fit onto the screen, and the hierarchy is animated to adjust its position. SpaceTree provides search and filtering functions to only display the subtrees on the path from the root to the node of interest. The user study shows that participants were confident with choosing the subtrees with large number of nodes, but often misinterpreted the width of the glyphs corresponding to the number of direct descendants rather than total leaves in the branch. This result reveals that the glyphs in SpaceTree are too abstract, failing to convey the structure details.

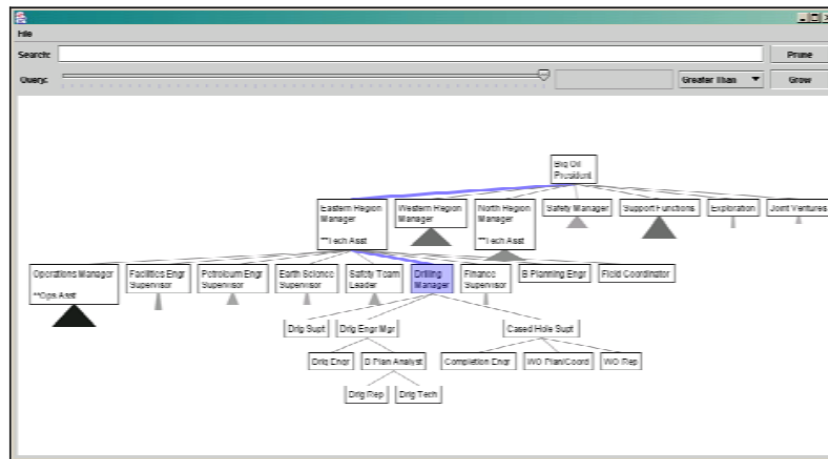


Figure 3.24: SpaceTree showing a hierarchy (Plaisant et al., 2002).

A similar approach is proposed by Heer and Card (2004). This approach extended original Degree-of-Interest (DOI) trees by employing multiple focuses and space constraint. As shown in Figure 3.25, uninteresting subtrees are collapsed into triangles, so that the tree can be arranged within a constrained area. Users are allowed to interactively explore by collapsing and expanding, and specify multiple interesting nodes. The algorithm progressively recomputes DOI values based on user interests and the space constraint, where nodes with lower interest values are automatically grouped into blocks if the screen space is not enough to show all the expanded subtrees entirely.

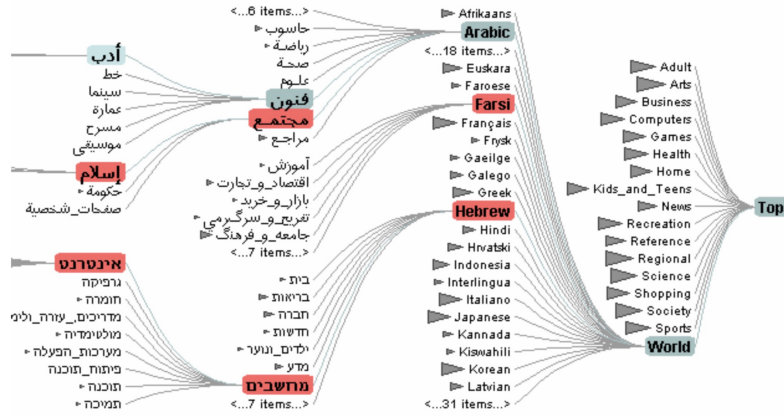


Figure 3.25: DOI Tree visualising a hierarchy with 600,000 nodes (Heer and Card, 2004).

### 3.4 Conclusion

Different ontologies and associations have various data volumes and structural features, requiring representative visualisations to address different issues and convey useful information. However, from the literature survey, it is clear that an effective, intuitive and easy to use visualisation tool is still missing. Existing approaches fail to show hierarchical structure effectively and achieve high scalability at the same time. The analyses on homogeneous and heterogeneous associations are also not supported well by current tools. Some of them introduce visual clutter when visualising large datasets. The interaction techniques such as zooming and panning that are employed to handle large ontologies cause a loss of context.

Employing appropriate combinations of visual marks, channels and alignments can enhance the effectiveness and scalability of a visualisation. Meanwhile, visual compression can be viewed as a promising strategy to tackle the complexity and scalability issues of visualisations. As all static representations have limits for the quantity of information that can be shown on a finite display space, when the limits are reached, interactions must be used to provide supplementary functionality for browsing and navigating through datasets.

To highlight the outcomes from this chapter, adjacency channel seems to be the best choice among node-link connection and area containment, with a good compromise of intuitiveness and scalability. Hybrid channel representations also provide another option, combining the best features of different channels, but may carry the drawback that they are less uniform and familiar to users. Therefore, exploring different alignments and possible hybrids of adjacency representations looks promising. Multiple inheritance in ontologies is one issue that needs to be addressed. Visual compression approaches such as structural clustering and motif simplification are also worth investigating for producing a compact and meaningful design to further improve the effectiveness and scalability of the visualisation. The interactions that can dynamically compress the items that are unwanted by users are proposed as a solution to explore and handle large ontologies, and avoid the issue of context loss caused by other interactive techniques.

## Chapter 4

# Visualising Large Ontologies

This chapter explores new approaches for visualising large ontology hierarchies. It firstly examines the design space of hierarchy visualisations. Then it describes the initial design for ontology visualisation in this research. The employed visual compression technique and interaction are also explored.

### 4.1 Design Space Exploration

The exploration of the design space for ontology visualisations starts from the approaches that are based on the layout of treemaps, which applies existing treemap variants to explore the data. Then icicle plot variants are investigated, resulting in a hybrid of icicle plot and treemap that draws from the strengths of both representations. The exploration process shows the evolution of the visualisation in this research. The pros and cons of each approach are also discussed. As discussed in Chapter 3, node-link connection based approaches suffer scalability issues. Also, a user study conducted by Fu et al. (2013) found that node-link visualisations fall short of showing ontology hierarchies constructively. As the hierarchy is important to ontology data analysis, node-link connection based visualisations are not considered.

#### 4.1.1 Nested Treemap

Initially, a nested treemap (discussed in Chapter 3) was utilised to visualise and explore the ontology data. An example of the nested treemap visualising an ontology is shown in Figure 4.1. The ontology classes are visualised as rectangles and the ontology hierarchy structure is represented by enclosing the child nodes within the parent nodes. The margins surrounding the child nodes depict the intermediate nodes.

Treemap representations demonstrate good scalability for visualising large hierarchies. Nested treemaps use margins between nodes in different levels to emphasise the hierarchical structure, which is one of the requirements for ontology visualisations (U3, U4, see Table 2.1).

The visualised ontology in Figure 4.1 is the Ontology of Adverse Events (OAE) (He et al., 2014), having 5606 classes and 18 levels. This ontology has a moderate size that

is not very large, but as can be seen in the figure, some nodes that are deep leaves or intermediate nodes are already too small to be shown. This could be solved by increasing the overall size of the treemap and prioritising the deepest nodes. However, with this approach, the overall size of the visualisation relies on the structure of the hierarchy and is hard to accommodate within the screen. While scrolling is inevitable to investigate treemaps, the path from a node to the root is hard to follow, as the containment approach might require scrolling both vertically and horizontally to trace a path.

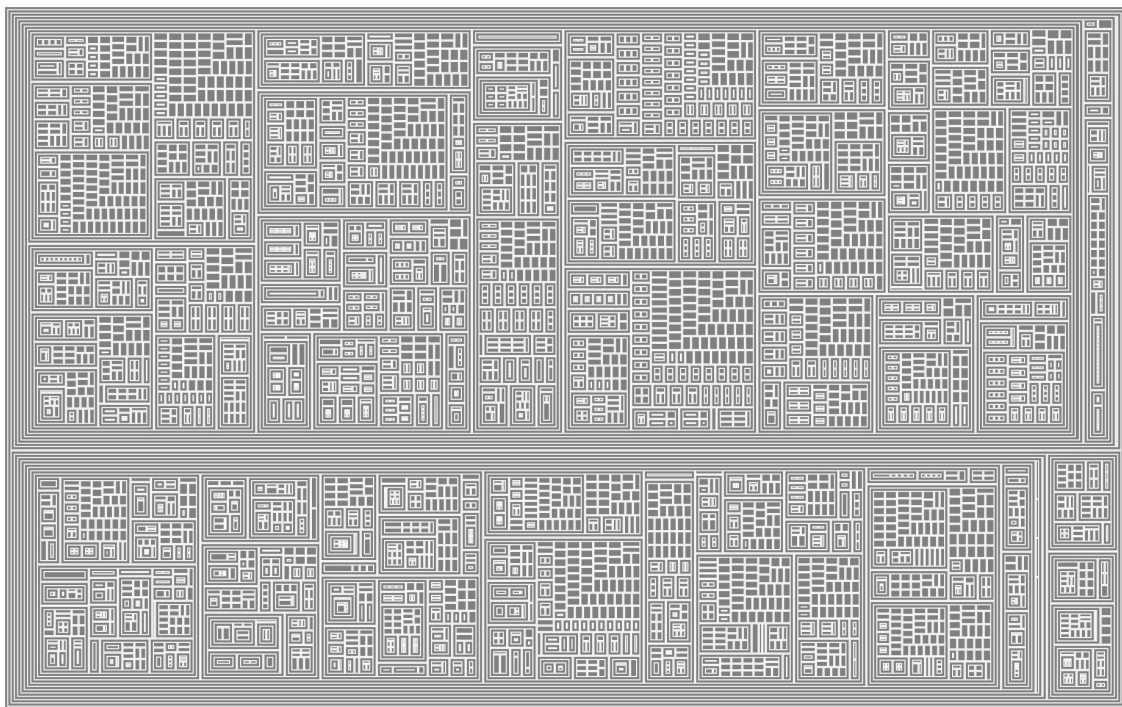


Figure 4.1: Nested treemap for the OAE ontology.

Another issue is, even though the treemap layout is trying to squarify the leaves that sets their height/width ratio as close as possible to one, some leaves are still larger than others, even though they have no greater importance. This problem has been proved to fall in the category of NP-hard problems by Bruls et al. (2000), which indicates it is not feasible to show all the leaf nodes in treemaps as squares.

#### 4.1.2 Sliced Treemap

To tackle the issues of the nested treemap, another representation named “sliced treemap” was created. Figure 4.2 shows three levels of the sliced treemap for OAE. In the sliced treemap, rather than being nested together, each level of the treemap is visualised separately as a “map”. The levels from the root to leaves are arranged from left to right. The relative position of boxes in different maps indicates the hierarchical relationships. For example, in Figure 4.2, the node highlighted in ① is the parent of the nodes highlighted in ②.

A sliced treemap can show the nodes on each level clearly. However, the hierarchical relationships are hard to interpret and it is difficult to map between different levels without

the assistance of any interaction, such as brushing. Also, this visualisation requires excessive space, which negates the original benefit of treemaps. As can be seen from Figure 4.2, only three levels can be fit in this figure if a reasonable space is given to leaf nodes. This is problematic given ontologies frequently have more than ten levels.



Figure 4.2: Three levels of a sliced treemap displaying the OAE ontology.

#### 4.1.3 Icicle Plot Variant

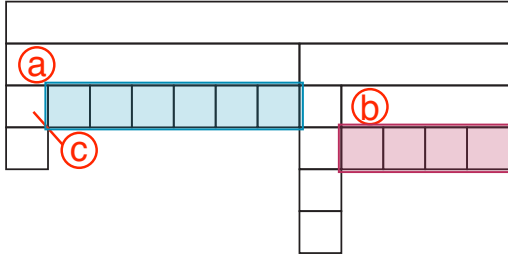
Another option was to take advantage of icicle plots (discussed in Chapter 3), which primarily emphasise the ontology hierarchy structure using boxes, where the children of a given node are displayed directly below the parent node, and the parent box’s width is the total width of all of its children (see Figure 4.3a). Icicle plots are more intuitive than treemaps, and can also show each level of hierarchy clearly.

##### 4.1.3.1 Hybrid Icicle Plot and Treemap

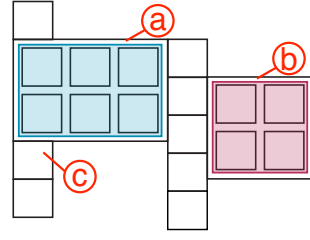
One disadvantage of icicle plots is the visualised hierarchy may not be too wide, i.e., have many leaf nodes, which is a common property of ontologies. In order to mitigate this disadvantage, the use of treemaps can be adopted in an icicle plot to accommodate the leaf nodes.

Figure 4.3b shows an example of this hybrid icicle plot and treemap, visualising the same hierarchy as in Figure 4.3a. In this hybrid representation, if a parent node has multiple leaf nodes, these leaf nodes are put into the parent node, following the layout of squarified nested treemaps. For example, as shown in Figure 4.3, node ① is the parent of the leaf nodes highlighted in the blue rectangle, and node ② is the parent of the leaf nodes highlighted in the red rectangle. The size of the “container” nodes ① and ② depends on the number of leaves inside them. The other nodes have a unified node size, no matter how many children they have and do not cover all their children as in icicle plots.

This hybrid representation reduces the overall width of the visualisation at the cost of a moderate increase in height. One major issue of it is that it is less uniform than either icicle plots or treemaps, potentially making it difficult to interpret the hierarchical structure. For example, in Figure 4.3, node ③ is a child of node ①, same as the leaf nodes highlighted in the blue rectangle. This can be easily identified with an icicle plot, while in



(a) Icicle plot.



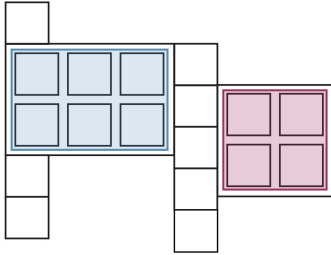
(b) Hybrid icicle plot and treemap.

Figure 4.3: Comparison of an icicle plot and the hybrid icicle plot and treemap.

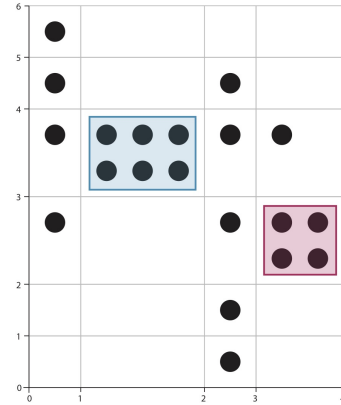
the hybrid representation, it needs mental effort to perceive the structure of the subtree under node @, since node © has a different arrangement to its sibling leaf nodes.

#### 4.1.3.2 GridPlot

In order to eliminate the inconsistency of the arrangement for leaf nodes and non-leaf nodes in the hybrid representation, another visual representation was explored, shown in Figure 4.4b, named “GridPlot”, visualising the same hierarchy as in Figure 4.4a. The blue and red rectangle highlight the same group of leaf nodes.



(a) Hybrid icicle plot and treemap.



(b) GridPlot.

Figure 4.4: Comparison of the hybrid icicle plot and treemap and GridPlot.

In GridPlot, each node is drawn as a dot rather than a rectangle as in the hybrid representation. The dots are arranged in a grid structure according to the following rule: starting from the root, put the root in the top left box; all the child nodes sharing the same parent are grouped inside the box under their parent's box; if a child has a descendant, i.e., it is not a leaf, it will be moved out from the group and put inside a new box on the same level as its siblings. As a result, the parent of a node is always either inside the box on top of that node, or inside the box on upper left of it. The size of each box depends on the number of nodes inside it. Like the squarified treemap arrangement for the leaf nodes in the hybrid representation (see Figure 4.4a), the grouped leaf nodes are arranged in a way to make their box as close to a square as possible.

Comparing to an icicle plot (see Figure 4.3a), the leaf nodes in GridPlot are placed more compactly, while comparing to the hybrid representation (see Figure 4.4a), the hierarchical structure of GridPlot is more clear and the visual elements are more uniform. The arrangement of leaf nodes in GridPlot addresses the matter that ontologies need lots of horizontal space and provides a possible solution.

## 4.2 OntoPlot

This section introduces the chosen design of the visualisation developed in this research, named “OntoPlot”, which benefits from the outcome of the design space exploration for visualising large ontology hierarchies. The section presents the employed visual compression technique, followed by the discussion of the design of glyph representations used for visual summary. The interaction functions are also described.

This section describes both a visualisation approach and an implemented system. The description here is the first version. The system is later revised, which will be discussed in the following chapters. The revised system is available at <http://ialab.it.monash.edu/ontoplot/>, with sample ontologies to try for non-ontology-experts. A demonstration video is provided in the link, illustrating the interface and functionality of the revised system. Also, the source code of OntoPlot is available on GitHub (<https://github.com/yingyangvis/ontoplot>).

### 4.2.1 Visual Representation

Figure 4.5b shows the basic visual style of the design. This design removes unnecessary segments in GridPlot that created visual noise and are likely to be misleading (see Figure 4.5a). The resulting visualisation (see Figure 4.5b) is similar in style to an icicle plot (see Figure 4.5c) and is named “OntoPlot”.

As discussed in Section 4.1.3, the use of a standard icicle plot to visualise an ontology hierarchy with many leaf nodes would not be ideal since the overall width of the visualisation would be proportional to the number of leaf nodes. To mitigate this, Wagner et al. (2018) used an approach that aggregates or removes the lower level nodes in an icicle plot representation and employed this icicle plot as a navigation panel to browse the hierarchically structured data. Conversely, OntoPlot shows the whole hierarchy. The nodes in OntoPlot are represented as circle glyphs within the boxes traditionally used in icicle plots. Where a number of a given node’s children are leaf nodes, these nodes are consolidated (wrapped) together in a single box that is taller and wider in order to accommodate multiple circle glyphs. A similar approach of wrapping leaves was identified in Graham and Kennedy (2007) which arranges leaf nodes in grids under the enclosure of their parent node. Even with the wrapping of leaf nodes, a large ontology may still be very wide. For this reason, the visualisation requires interaction to easily navigate and scroll horizontally through the entire ontology (U5, see Table 2.1).

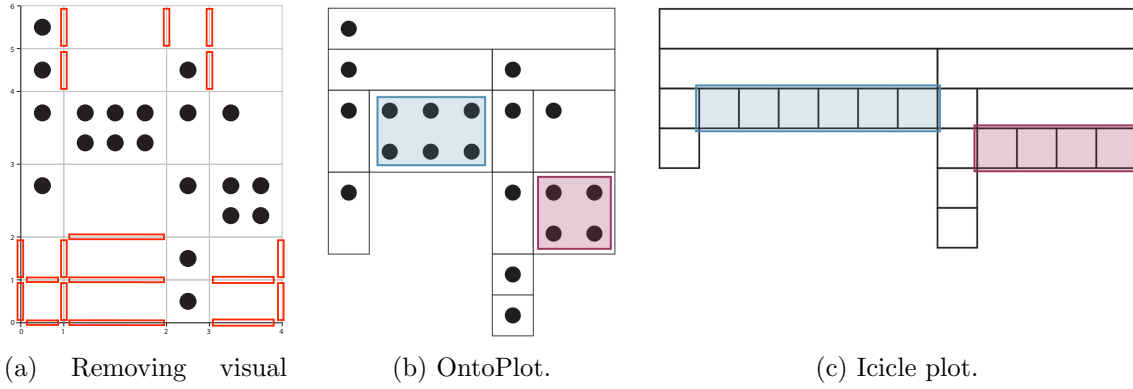


Figure 4.5: Comparison of GridPlot, OntoPlot and icicle plot.

### 4.2.2 Visual Compression

As discussed in Chapter 2, an essential need of large ontology visualisations is being able to find and focus on *interesting* information from the ontology hierarchy (U1, U2, see Table 2.1).

While considering the *interesting* information and the class to which it applies, there will often be a large subset of the ontology which is *uninteresting* in terms of the task at hand (e.g., classes pinned or searched by users (as discussed in Section 5.3.2) or classes having associations (as discussed in Section 5.1.1)). For this reason, a form of visual compression (discussed in Chapter 3) that compresses the *uninteresting* subtrees is employed to give more prominence to the *interesting* parts of the hierarchy.

#### 4.2.2.1 Glyphs for Visual Summary

Three cases worthy of compression are identified. The parts of the ontology being compressed are replaced with three compression glyphs, using the motif simplification approach discussed in Chapter 3 (see Figure 4.6):

- **Leaf nodes:** Where an interesting node has multiple uninteresting leaf nodes as children, these nodes will already be shown as a number of circle glyphs in a single box. These multiple circles are replaced with a single square glyph.
- **Chain:** Where an interesting node has a descendent subtree that is a chain of only uninteresting nodes, this chain is replaced with a single box containing a thin block glyph.
- **Subtree:** Where an interesting node has a descendent subtree that contains only uninteresting nodes and does not fall into the previous two categories, the subtree is replaced with a single box containing a triangle glyph.



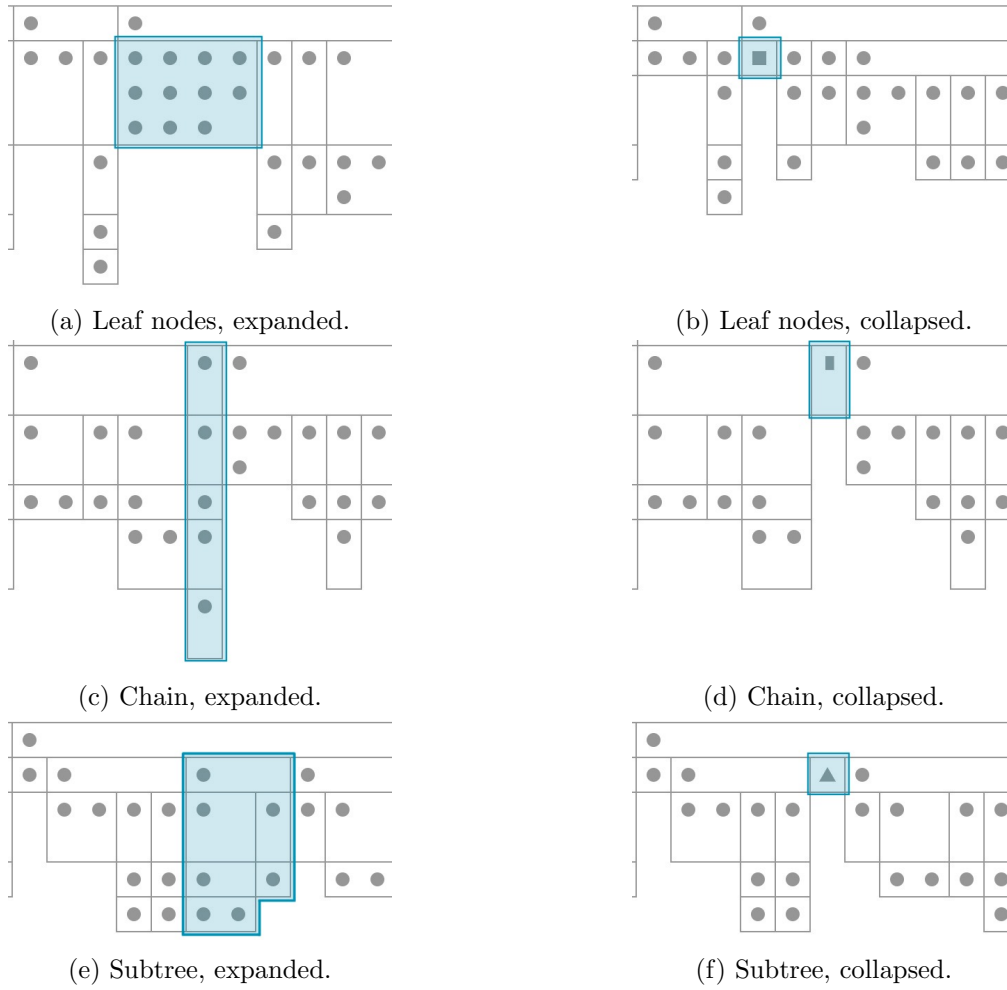


Figure 4.6: Examples of visual compression.

Given the classes that are *interesting*, any subtree in the hierarchy can be considered *interesting* or *uninteresting* depending on whether it contains any *interesting* classes. Using this, the hierarchy is walked through to get the set of nodes (classes) that can be compressed as described above. As shown in Algorithm 1, a recursive depth-first traversal of the tree is performed from the root node, where for each node the recursive call returns the number of interesting nodes in the subtree and an array of any collapsible nodes. If the node is *interesting*, it adds its children that are *uninteresting* to the collapsible node array. If the node is *uninteresting* along with its children, it replaces the collapsible node array with just itself (collapsing it would hide the entire subtree). The second parameter of the function in the algorithm is passed by reference and returns a value. The function is called with the root node of the hierarchy and when it returns it will have populated the array given in the second argument with the root of each *uninteresting* subtree. This algorithm gives a set of nodes that are at the root of each subtree as collapsible nodes. Since these collapsible nodes can be discovered in a single depth-first traversal, this process is linear in the number of classes.

---

**Algorithm 1:** Get *uninteresting* parts of a hierarchy.

---

```

uninterestingParts(node, nodeArray)
1  Initialise empty array childNodeArray, variable count = 0;
2  if node.interesting then
3    | count = 1;
4  foreach child in node.childNodes do
5    | Initialise childCount = uninterestingParts(child, childNodeArray);
6    | count += childCount;
7  if count = 0 then
8    | Add node into nodeArray;
9  else
10   | foreach element in childNodeArray do
11     | | Add element into nodeArray;
12 return count;

```

---

#### 4.2.2.2 Design Space of Glyphs

As discussed in Chapter 3, visual summary glyphs are used to replace common patterns in the data, for the purpose of raising the visibility of common structures and reduce the visualisation complexity.

There are different design settings for the glyphs: contour only, structure only, contour + structure and metaphor (Ward, 2002). Figure 4.7 draws examples of each of them for the three cases discussed in Section 4.2.2.1 (leaf nodes, chain and subtree) that are worthy of compression.

The glyphs in this research use the contour setting. The reason for this is that it is more consistent with the circle glyphs that represent classes in the visualisation than the structure setting. Also, the contour setting is simpler than the contour + structure setting and metaphor setting, which gives a better choice for large ontology visualisations that might consist of many visual elements, and it is easier to read when shown at a small size.

#### 4.2.3 Interaction

This section describes the interaction supporting the investigation and exploration of ontologies.

##### 4.2.3.1 Collapse and Expansion

While the visual compression is performed automatically to hide the uninteresting parts of the ontology hierarchy, in an interactive visualisation the subtrees could always be expanded and collapsed interactively in order to show or hide parts of the ontology (U5, see Table 2.1).

Double-clicking on the glyph of a compressed part of the tree (square, thin block or triangle) can expand a subtree. A subtree can also be compressed by double-clicking on the circle glyph corresponding to the root of that subtree.

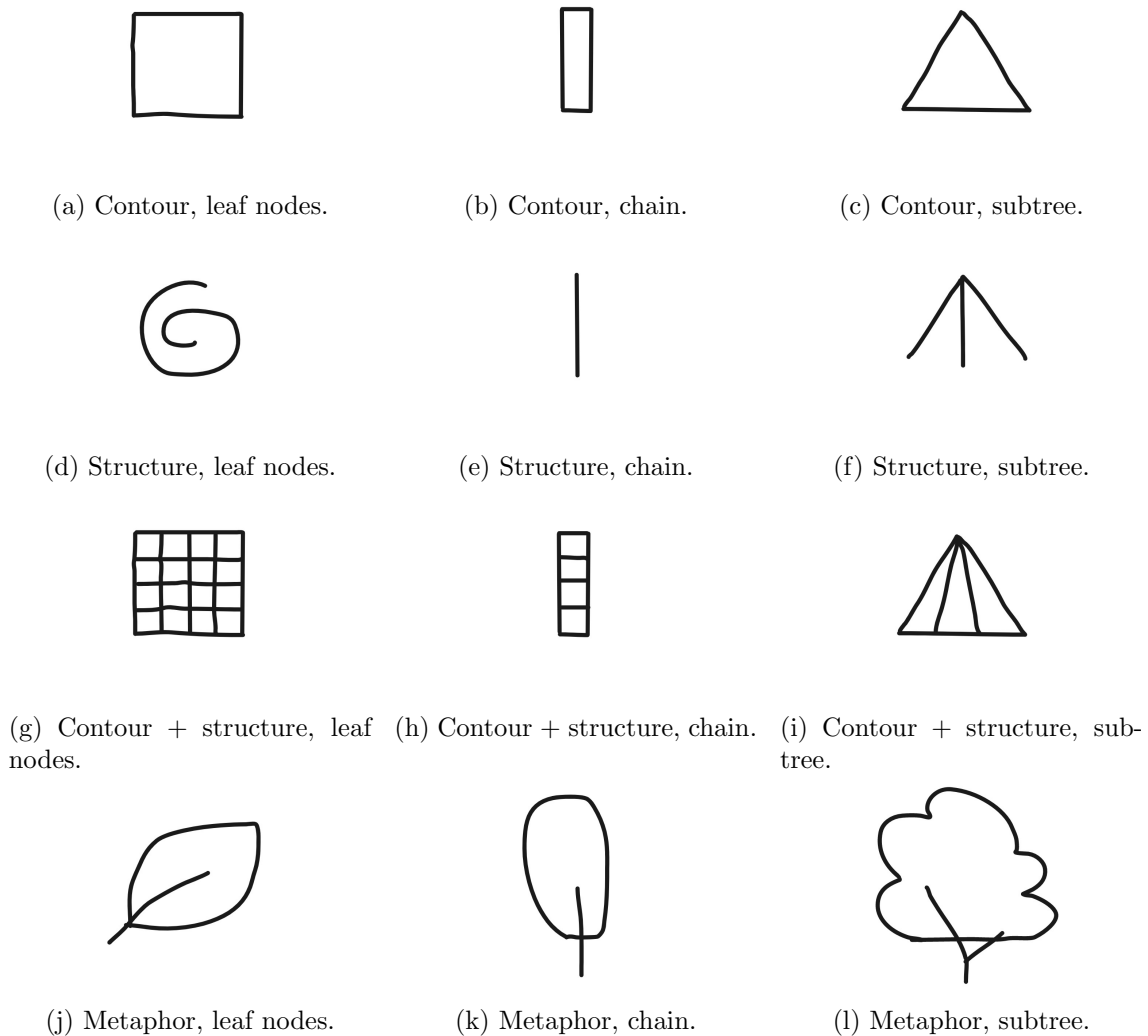


Figure 4.7: Sketch of different settings of glyphs for compressed hierarchy subtrees.

Interactive collapsing or expanding operations are performed efficiently without recomputing and redrawing the entire visualisation. Instead, the system gets the position of the double-clicked node and retrieves the parts of the visualisation that are on the right and below this node. The positions of all of these parts are recomputed based on the expanded or collapsed region in a linear pass over each element, and then they are redrawn.

#### 4.2.3.2 Pinning Class Label

The class labels are not displayed by default since these take up a large amount of space and can cause problems with occlusion when densely packed, but they sometimes need to be displayed on the visualisation (U1, see Table 2.1).

Class labels can be shown by shift-clicking class glyphs if users are interested in particular classes. The shift-clicking pins the class labels on top of the clicked class glyphs (see Figure 4.8). This allows interesting classes to be marked and easily spotted in the ontology hierarchy.

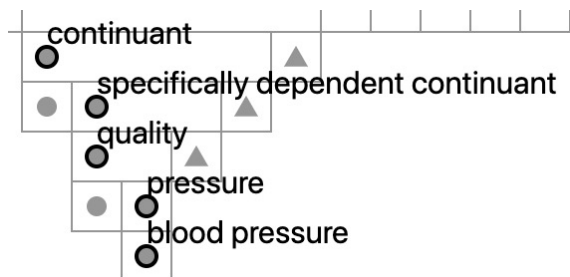


Figure 4.8: An example of pinning a series of class labels from the ancestors to a leaf class of interest.

### 4.2.3.3 Search

The visualisation offers a search field. When a term is entered in the search field, a scrollable list of matching classes in the ontology will be displayed (see Figure 4.9 (a)). The matching classes are highlighted in the ontology hierarchy with an additional circle drawn around their class glyphs (for example, see Figure 4.9 (b)).

Selecting a class from the matching classes list scrolls the view of the visualisation to make that class visible. Also, the class label of the selected matching class is shown on the top of its class glyph (see Figure 4.9 (c)). If the matching class happens to be compressed in a compression glyph, the selection will expand the compression glyph to show this class.

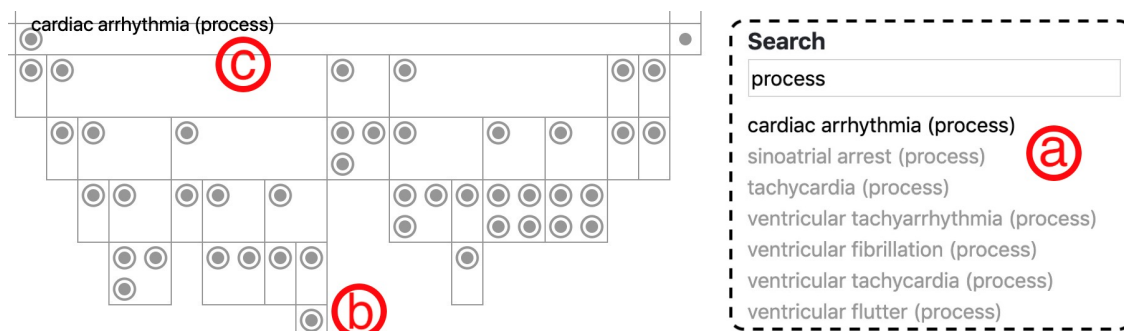


Figure 4.9: When the user enters the search term “process” and selects the class *cardiac arrhythmia (process)* from the matching classes list, the label of the *cardiac arrhythmia (process)* class is displayed. As all the classes in the *cardiac arrhythmia (process)* subtree contain the term “process”, their glyphs are surrounded by additional circles. Note, the search panel consisting of the search field and the matching classes list is cropped and brought together in this image.

## 4.3 Conclusion

This chapter described the attempts using treemaps and icicle plots for visualising large ontology hierarchies. It presented the OntoPlot visualisation, which draws on the advantages of the representation of treemaps and icicle plots to seek a balance between the width and the height of the visualisation. OntoPlot clearly shows the ontology hierarchy structure and improves the space-efficiency of the visualisation. It employs visual compression technique and uses visual summary glyphs to give to the interesting information

in ontologies and reduce the visual structural complexity. It also offers interactivity to explore ontology hierarchies.

The next chapter will discuss the process of visualising homogeneous associations using OntoPlot. It will discuss a prototype system based on the approaches presented in this chapter, a user evaluation of the prototype, resulting revisions to the system, and finally an expert user evaluation.



## Chapter 5

# Visualising Homogeneous Associations

The last chapter discussed the visualisations for ontology hierarchies and presented the design of OntoPlot. This chapter describes the work for visualising homogeneous associations in ontology hierarchies (involving one property), building on the work described in the last chapter.

In this chapter, a prototype OntoPlot visualisation for homogeneous associations is demonstrated and evaluated. A refinement of the prototype, based on the results and feedback of the evaluation, is described. An expert user study that tests the usability of the refined OntoPlot is then presented. The refined OntoPlot and expert user study are described in (Yang et al., 2019). Finally, some extensions to OntoPlot after the expert user study are described in the last section of the chapter.

### 5.1 Prototype

This section discusses the prototype of OntoPlot for visualising homogeneous associations involving a single property. It firstly describes the visual interface and then presents the interaction supported by the visualisation.

#### 5.1.1 Visual Interface

Figure 5.1 shows the visual interface of the prototype.

When an ontology is loaded for visualisation, a list of object properties found within the ontology is presented (see Figure 5.1 ①).

When a property is selected, the classes that have associations with the selected property are treated as *interesting*, and the parts of the hierarchy not containing any interesting classes will be considered *uninteresting*. The system detects the *uninteresting* parts and uses the visual compression technique introduced in Chapter 4 to collapse them within the ontology. For example, Figure 5.1 ② shows a subtree compressed as a triangle.

Then the associations are visualised by colouring the circle glyphs of classes to which the associations apply (for example, see Figure 5.1 ③).

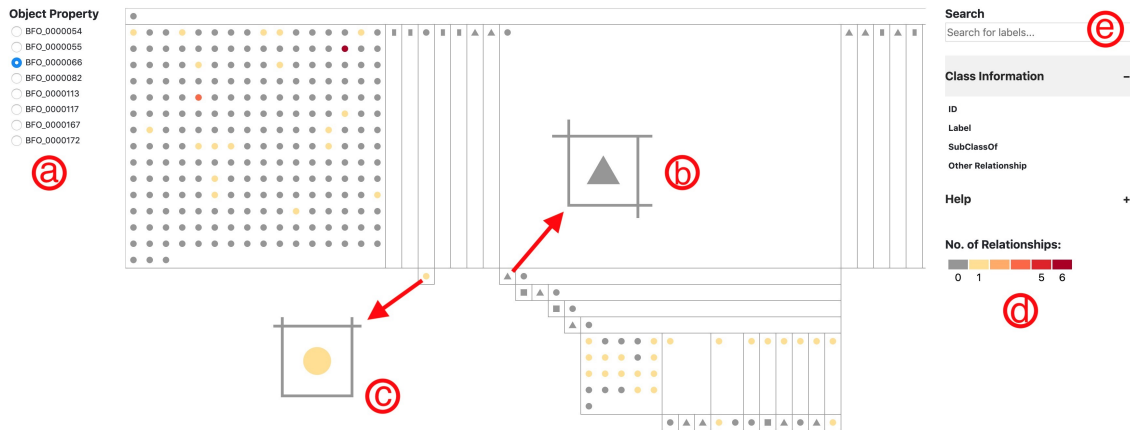


Figure 5.1: The visual interface of the OntoPlot prototype. ① is the property list. ② is an example of a subtree compressed as a triangle. ③ is a class that has associations. Classes with associations are highlighted in colour based on the key ④ on the right-hand side. Note, the search field ⑤ introduced in Section 4.2.3.3 is shown on the top-right.

#### 5.1.1.1 Colour Key

A range of colours is used, where intensity signifies the number of associations applying to that class. The colour key is dynamic depending on the maximum number of associations of the selected property applying to any class (see Figure 5.2).

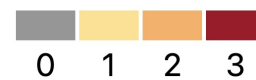
Nodes with the minimum and maximum number of associations are separately coloured (see Figure 5.2a). To emphasise the maximum value, the colour dark red is always used to draw the circle glyphs with the highest number of associations (U7, see Table 2.3). Further colours are used to categorise values interpolated between these. If the maximum value is less than or equal to six, each intermediate value is represented as discrete colour values in the colour key (see Figure 5.2b).

#### No. of Associations:



(a) When the maximum value is greater than six, the intermediate colours are shown as a range.

#### No. of Associations:



(b) When the maximum value is less than or equal to six, the intermediate colours are given discrete values.

Figure 5.2: Colours for associations. Unique colours show the minimum and maximum number of associations.

#### 5.1.2 Interaction

This section describes the interaction facilitating the exploration of association information in ontologies.



### 5.1.2.1 Scrolling

Although visual compression techniques are employed to reduce the visual elements that are *uninteresting*, the visualisation will frequently be wider and occasionally taller than the screen. Also, any compressed parts can be interactively expanded. Thus, the visualisation supports scrolling horizontally and vertically to browse the entire ontology.

### 5.1.2.2 Information Pop-ups for Classes

Initially, the information of associations is not displayed in text form. Instead, the visualisation displays the class ID, class label and the number of associations on a class in a pop-up window while the glyph corresponding to a class is being hovered over (U6, see Table 2.3), as shown in Figure 5.3.

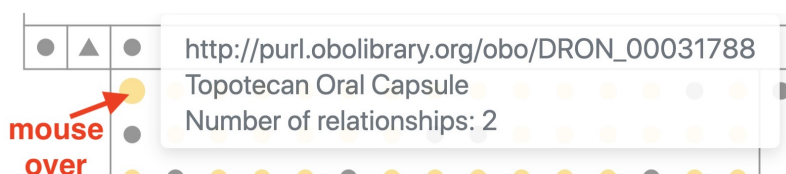


Figure 5.3: A pop-up window shows association information while hovering over classes.

### 5.1.2.3 Class Selection

Classes can be selected by clicking on them to investigate individual associations. When a class is selected, the visualisation updates to colour only the classes that the selected class has associations with (U8, U9, see Table 2.3), as shown in Figure 5.4 (a). The *interesting* and *uninteresting* parts are recomputed. The visualisation visually compresses the *uninteresting* parts of the ontology hierarchy and only shows associations that the selected class is involved in. The selected class is given a black outline and its class label is displayed (see Figure 5.4 (b)).

As discussed in Chapter 2, ontology hierarchies usually contain multiple inheritance. OntoPlot addresses this issue by duplicating child classes under each of their parent classes. If the selected class has been duplicated, i.e., it appears multiple times in the hierarchy, these identical classes will be coloured green.

While a class is selected, the right-hand panel of the visualisation displays additional information on that class, including a textual list of all its associations (U6, see Table 2.3), as shown in Figure 5.4 (c).

Deselecting a class is performed by clicking on an already selected class. When a class is deselected, the visualisation goes back to its original state of showing all the associations with the selected property.

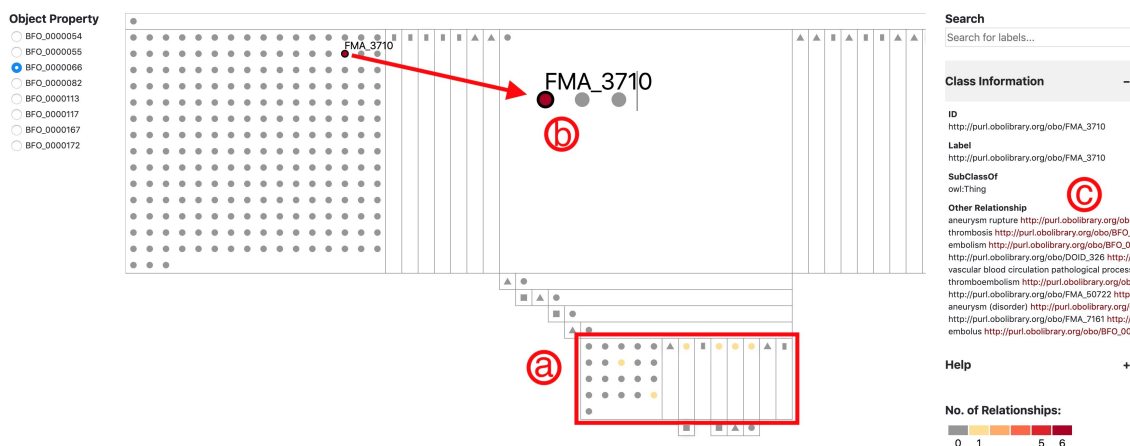


Figure 5.4: The visualisation when the class FMA\_3710 is selected.

## 5.2 Prototype Evaluation

In order to determine if the design of the prototype supports the use cases and meets the design requirements for visualising large ontologies and their homogeneous associations discussed in Chapter 2, a user-based evaluation was conducted.

21 participants were recruited, including 2 domain experts and 19 general users.

### 5.2.1 Study Design

In the user study, the prototype of OntoPlot was compared with Protégé (Noy et al., 2000).

The reason of choosing Protégé is because it is a robust tool. Protégé is the most widely used and actively maintained tool for ontology creation and editing in the ontology engineering community (based on citations). It provides a baseline representation—an indented list—for ontology hierarchy browsing and visualises non-hierarchical associations as text lists in separate views (see Figure 5.5).

Also, as mentioned in Chapter 2, domain experts frequently use Protégé to perform their ontology-based analysis, and also present their work using screenshots of the Protégé indented list view with manually added annotations to indicate the association strength in hierarchies (see Figure 2.5).

Protégé is a fully-featured ontology engineering environment, with many panes, views and functionality not necessary for the study. To avoid confusing the participants with a complex interface, Protégé was simplified by removing the unnecessary items from the interface, such as “Data properties” and “Individuals” panes, and the “Class Annotations” views. Some check boxes in the views and search window were also deselected to avoid irrelevant information being shown to participants. To better support the tasks, the interface layout of Protégé was modified to avoid view switching. The “Object properties” pane and the “Classes” pane were positioned side-by-side, the “Class Description” view and the “Class Usage” view were placed next to each other on top of the “Property Usage” view. Figure 5.5 shows the interface layout configured for the study.

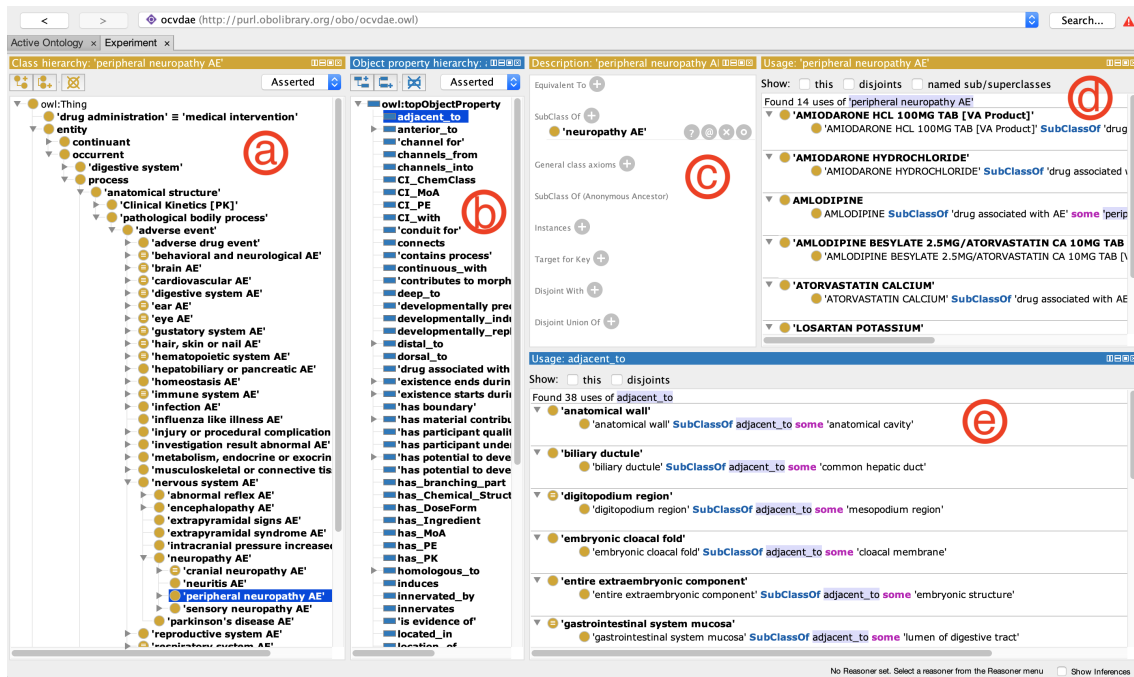
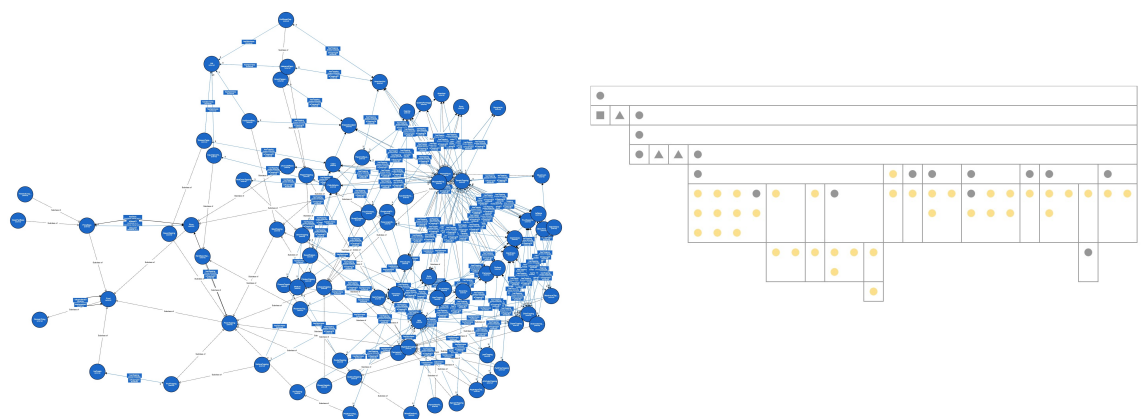


Figure 5.5: Protégé interface as configured for use in the study. ①: Class pane, ②: Object Property pane, ③: Class Description view, ④: Class Usage view, ⑤: Property Usage view.

### 5.2.2 Comparison with WebVOWL

As mentioned in Chapter 3, several tools support the display of non-hierarchical associations alongside an ontology’s inheritance hierarchy. The suitability of these tools was investigated for comparison in the study. The Protégé visualisation plug-ins Jambalaya (Storey et al., 2001) and Knoocks (Jurcik, 2012) are no longer maintained and do not work with the current version of Protégé. Neither OntoViewer (da Silva et al., 2012) nor the three-view tool described in (Kuhar and Podgorelec, 2012) are publicly available.

WebVOWL (Lohmann, Link, Marbach and Negru, 2014), which visualises ontology hierarchical and non-hierarchical relationships using a connection-based style, has also been attempted. Figure 5.6 shows the comparison between WebVOWL and OntoPlot visualising the Pizza ontology (Drummond et al., 2007). The Pizza ontology in the visualisations has 101 classes without duplication for multiple inheritance. As WebVOWL does not support filtering relationships based on classes or properties, Figure 5.6a visualises all 148 associations in this ontology, while Figure 5.6b shows one property selected resulting in 19 associations in OntoPlot. As can be seen from the comparison, WebVOWL does not differentiate hierarchical and non-hierarchical relationships, thus the hierarchy structure is not depicted as clearly as in OntoPlot. OntoPlot also better emphasises *interesting* associations in the ontology hierarchy. However, WebVOWL does take the advantage of node-link connections such that it can avoid duplicating classes under each of their parents to indicate multiple inheritance, which is the disadvantage of OntoPlot. In addition, the target and source classes of the relationships can be directly determined from the visualisation in WebVOWL without any interactions.



(a) The Pizza ontology visualised in WebVOWL. All classes and associations are shown. (b) The Pizza ontology visualised in OntoPlot. Based on the selected property, the *interesting* classes are shown and highlighted. Other parts are visually compressed.

Figure 5.6: Visual comparison between WebVOWL and OntoPlot on the Pizza ontology. Note, as the Pizza ontology does not contain multiple inheritance, there are no duplicated classes in OntoPlot.

Although the node and links can be followed in the Pizza ontology with WebVOWL, when attempting to visualise an ontology of the size required for the user study, the readability of the visualisation in WebVOWL reduced dramatically. As discussed in Chapter 1.1, one obvious drawback of node-link connection is the resulting “hairball” effect when encountering large ontologies. Figure 5.7 shows an example of the visualisations of a larger ontology generated by WebVOWL. The investigation was done using the ODNAE ontology (Guo et al., 2016), which has 1,545 classes and 810 associations. As can be seen from the visualisations, it is not easy to identify and trace the hierarchy structure and the non-hierarchical associations, either when zooming in for detailed view (see Figure 5.7a) or when zooming out for overview (see Figure 5.7b). Moreover, WebVOWL only supports ontologies (OWL format) that are less than 5MB and the ontology OCVDAE (Wang et al., 2017) with 2,949 classes and 8,308 associations required by the tasks in the user study is beyond this boundary (OCVDAE 6.7MB, ODNAE 2MB). Thus WebVOWL is not a suitable and a comparable tool to OntoPlot for the user study.

### 5.2.3 Tasks

As discussed in Chapter 2, a range of important use cases and user needs for biomedical ontologies were identified from the literature as well as from interviews with the domain expert. To test the usability of the prototype with respect to the identified user needs, ten tasks were designed from the use cases and organised into three groups, shown in Table 5.1.

The first group of tasks (G1) focuses on the hierarchical structure of ontologies. While these are basic hierarchy comprehension tasks, they are essential to almost all analyses of

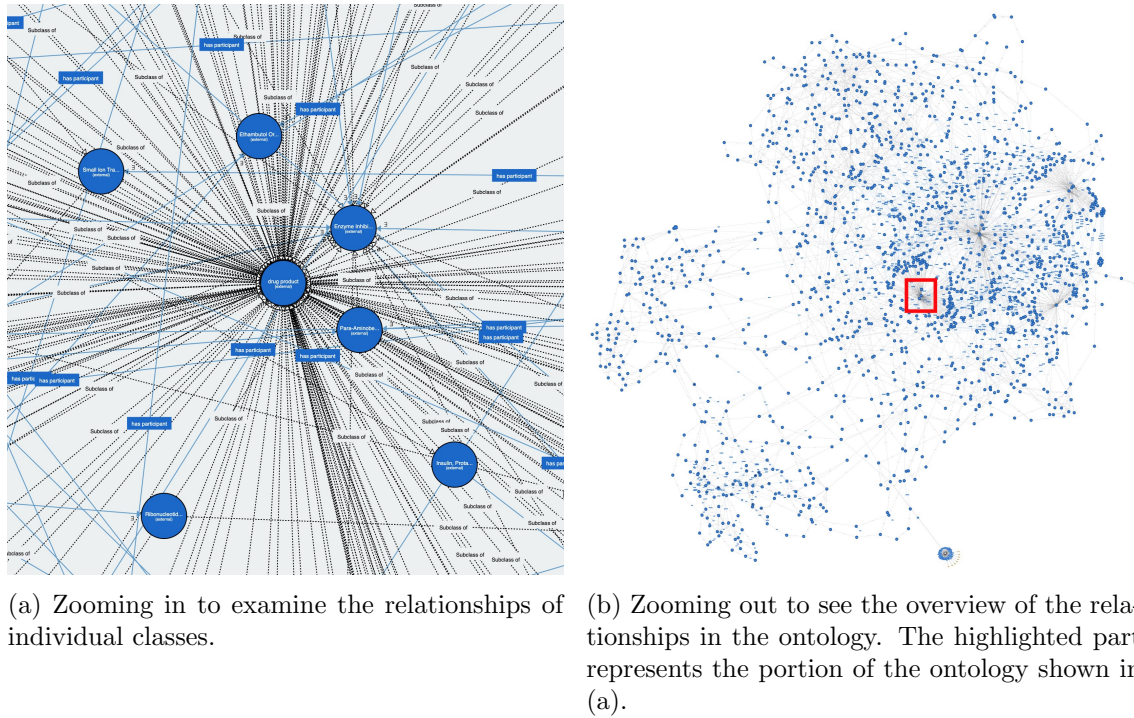


Figure 5.7: When visualising the ODNAE ontology with 1,545 classes and 810 associations in WebVOWL.

ontologies. For example, tasks T1, T2, and T3 ask about parent-child relationships, requiring exploration of the ontology hierarchical structure (U5), and they investigate whether the visual compression and glyphs in the prototype impact the cognition of the ontology hierarchy. Similarly, T4 asks participants to trace the hierarchical path from a class to the root which is related to generalising concepts (U3). T5 examines the intersection of two subtrees supporting common knowledge discovery (U4).

The second group of tasks (G2) focuses on non-hierarchical associations. Both T6 and T7 require an exploration of the associations for a class (U6). T6 asks for all classes associated with a class, while T7 asks for the total number of them. T8 requires participants to find the class with the highest number of associations in the ontology, which identifies significant classes (U7).

The third group of tasks (G3) further examines the associations together with the hierarchical structure. These tasks are the most complex ones but essential for analysing associations on the class level. T9 asks for the parent having the most children with associations, which helps determining the *class effect* (U8). T10 finds the outlier (class without associations) among a group of sibling classes with associations, providing evidence for predicting undiscovered associations (U9).

#### 5.2.4 Hypotheses

The prototype was hypothesised to perform similarly to Protégé for G1 hierarchy tasks (H1), since both tools clearly emphasise the hierarchical structure of ontologies. The prototype was expected to outperform Protégé on G2 association tasks (H2) and G3

Table 5.1: Tasks in the experiment.

Group	Task	Use case	Description	Example instruction
G1. Hierarchy	T1	U5	Identify the parent of a class.	Please tell me the parent of “skin of body”.
	T2	U5	Identify the child(ren) of a class.	Please tell me the children of “limb segment”.
	T3	U5	Identify the sibling(s) of a class.	Please tell me the siblings of “anatomical space”.
	T4	U3	Identify the path from a class to the root.	Please tell me the path from “process” to the root.
	T5	U4	Identify the closest common ancestor of two classes.	Please tell me the closest common ancestor of “anatomical collection” and “anatomical surface”.
G2. Association	T6	U6	Identify the classes associated with a class.	Please tell me the classes which have the “may_prevent” association with the “Pain” class.
	T7	U6	Identify the number of associations of a class.	Please tell me the number of “may_prevent” associations of the “Hypertrophy” class.
	T8	U7	Identify the class having the highest number of associations.	Please tell me the class which has the most “may_treat” associations.
G3. Hierarchy + Association	T9	U8	Identify the parent class with the most children who are associated with a class.	Please tell me the class which has the most children that have the “adjacent_to” association with the “full formed stage” class.
	T10	U9	Identify a class that is not associated with a specified class, but all of its sibling(s) are associated with that class.	Please tell me the class whose siblings all have the “site_of_metabolism” association with the “Channelopathy” class, but that class itself does not have such an association.

hierarchy and association combined tasks (H3), since it is designed to support ontology association analysis.

### 5.2.5 Datasets

Two biomedical ontologies CVDO (Barton et al., 2014) and OCVDAE (Wang et al., 2017) with sufficiently different sizes were used for the study, so that the evaluation could be conducted on two difficulty levels. CVDO (518 classes) was chosen as a small dataset, and OCVDAE (4,589 classes) was chosen as a large dataset. In total, there are 8 object properties and 551 non-hierarchical associations in CVDO. In OCVDAE, there are 118 object properties and 20,269 non-hierarchical associations. In order to keep the experiment to a reasonable time, the classes with less than 25 associations were selected to ask questions about.

In addition, a training ontology was created to introduce the tasks to the participants. As most participants were not expected to have experience with ontologies, the training ontology was kept simple and small. The training ontology contained 15 classes, 2 object properties, and 6 associations, but covered all possible situations in the larger ontologies used in the study.

### 5.2.6 Procedure

Initially, a within-subjects design was used for the experiment: 2 tools  $\times$  2 ontology sizes  $\times$  10 tasks (+ training). One pilot test was run, and it revealed that the experiment took too long to cover all tasks in the 2 ontologies. Then a decision was made to split the tasks into two task sets and balance the ontology size and task type in each set. The final design was 2 tools  $\times$  2 ontology sizes  $\times$  5 tasks (+ training) where each participant performed all 10 tasks, but these tasks were split across the two ontologies (5 tasks each).

Participants performed tasks using the same ontology with different tools. To avoid issues of memorisation, all class and object property labels were systematically renamed and shortened to be different but the same length when used in each tool.

The order of tasks for each tool was fixed, but the order of tools shown to different participants was counterbalanced.

Before the experiment, participants were asked to answer some questions regarding their background knowledge and experience with ontologies, visualisation and ontology visualisation.

Participants were required to complete training before performing the experimental tasks. They were firstly shown an introductory video to explain the basic concepts of ontologies. Participants also completed training on each tool before using each of them. They were shown introductory videos to demonstrate the interface and functions of the tool, and were then required to use the tool to answer 10 sample questions with the training ontology. The sample questions covered all experiment tasks in order to allow participants to be familiarised with the tools and the tasks. While answering the sample questions, participants were guided to practise the functions that were needed in the actual tasks for each tool, such as searching, clicking classes or object properties, double-clicking to



expand or collapse subtrees, hovering the mouse cursor over classes to read class labels and association information, marking classes by pinning labels on them in the prototype, and going back or forward in Protégé. After each training question, participants were shown the correct answer, and an explanation was given if they did not answer correctly. In this case, the participants would be asked to answer the same questions again with different data.

A website was developed to guide participants through the study, give them access to training tasks, tasks and survey questions, and also to collect participants' answers and record completion time for each task. When participants were ready to begin a task, they clicked a button to load that task. When they finished a task, they clicked a button to indicate they had completed this task and were ready to progress to the next task. As switching object properties in the large ontology (OCVDAE) in the prototype required a couple of seconds to load the visualisation, the loading time was excluded from the task completion time. To keep the experiment within a reasonable time, a time warning was triggered for each task at the 2-minute point. For any task, if participants found it too difficult to answer, they could choose to skip that task.

After completing the tasks for each tool, participants were asked to complete a survey, rating the difficulty level and the confidence level of their answers for each group of tasks. Participants' preferences and comments were also collected at the end of the experiment. As Protégé is an existing system, participants were also asked whether they had used Protégé before the experiment.

Each experiment session lasted approximately one hour, including training and surveys.

### 5.2.7 Participants and Apparatus

The 21 participants included 2 domain experts, and 19 university students and academics. Of these, 8 were female, and 13 were male. Their age ranged from 21 to 61. All participants had normal or corrected-to-normal vision, and none were colour-blind.

Of the 21 participants, 8 participants had experience using visualisations and 5 had experience in developing visualisations. 7 participants had experience using ontologies, and 3 had ontology development experience. Among all the participants, only the 2 domain experts had used Protégé, while another 2 participants had used other ontology-based systems like Unified Medical Language System (UMLS) and tools developed by the Gene Ontology Consortium.

The participants recruited from the university used a 1.6 GHz Intel Core i5 laptop with 4GB of RAM, using a 24-inch monitor with a resolution of 3840x2160 pixels. The domain experts did the experiment remotely, using their own laptops at a resolution of 1600x900 pixels. For the remote participants, their experiments were observed via video calls.

### 5.2.8 Results

The results of one participant were considered to be invalid. In most of the tasks the participant did not use the correct class referred to in the question, and sometimes waited



for the time warning message to appear and then randomly chose a class as the answer. By excluding this participant therefore 20 valid samples were used for the analysis.

Accuracy and completion time were measured for each task. Difficulty level, confidence level, preference ranking and learning effort (as rated by participants) were collected.

As the data was not normally distributed, the non-parametric *Wilcoxon test* was used to compare accuracy between the two tools (Field et al., 2012). For the completion time data, only the time for correct answers was used. Therefore, the non-parametric *Whitney-Mann test* was used for unequal samples (Niroumand et al., 2013). For the rated results, *Wilcoxon test* was also used to analyse significance.

**Accuracy.** Figure 5.8 shows the details of mean accuracy for each tool per task, and Table 5.2 contains the accuracy *Wilcoxon test* results.

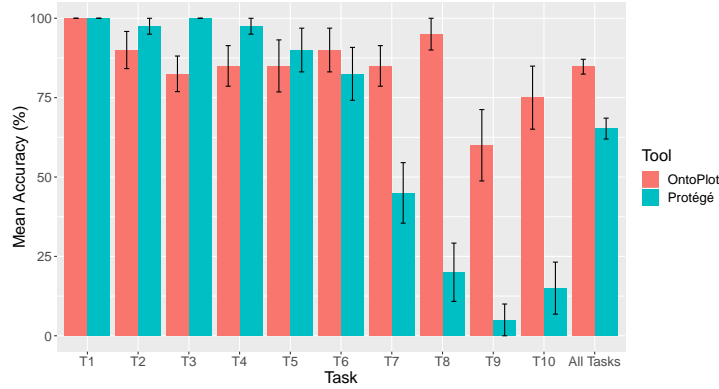


Figure 5.8: Mean accuracy for each tool per task.

Table 5.2: Summary of statistical significance (Wilcoxon test  $p$  values) of accuracy difference as summarised in Figure 5.8, for both ontologies and each of the two individual ontologies.

Task	Overall	CVDO (small)	OCVDAE (large)
T1	1	1	1
T2	0.34470	0.34580	1
T3	0.01788*	0.18140	0.07186
T4	0.11980	1	0.08897
T5	0.76560	0.34580	0.77280
T6	0.57080	0.37110	1
T7	0.00943**	0.00476**	0.21860
T8	0.00012***	0.00190**	0.03689*
T9	0.00260**	0.01966*	0.07260
T10	0.00063***	0.01966*	0.01966*
*** $p < 0.001$ ** $p < 0.01$ * $p < 0.05$			

For most of the Hierarchy tasks (G1) both tools achieved high accuracy, with Protégé having marginally better performance. One exception is for T3 (examining siblings), there is a significant difference ( $p < 0.05$ ) between the prototype and Protégé, with Protégé performing better than the prototype. The test also reveals the prototype outperformed Protégé for the Association tasks (G2) which asked for counting and ranking associations

(T7, T8), with significant differences ( $p < 0.01$  and  $p < 0.001$ ) between the prototype and Protégé. Highly significant differences were also found for both tasks (T9, T10) in Hierarchy + Association (G3), with Protégé performing worse than the prototype. These results were not affected by ontology size, except in T10 where the accuracy of both tools on the large ontology dropped by 30% compared to the small ontology.

**Completion Time.** Results for completion time are shown in Figure 5.9, and the significance of time resulted from the *Whitney-Mann test* can be found in Table 5.3.

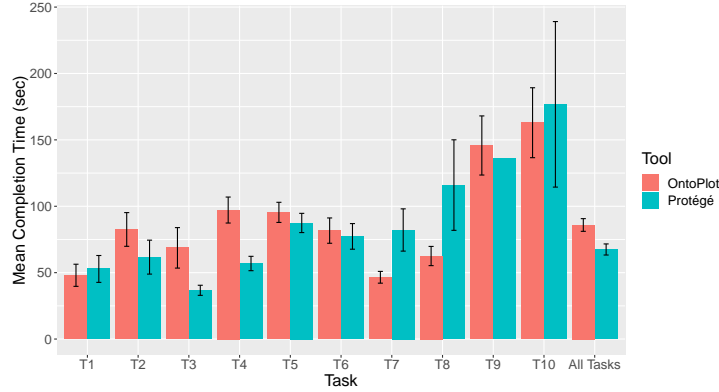


Figure 5.9: Mean completion time for each tool per task.

Table 5.3: Summary of statistical significance (Whitney-Mann test  $p$  values) of differences in completion time as summarised in Figure 5.9, for both ontologies and each of the two individual ontologies.

Task	Overall	CVDO (small)	OCVDAE (large)
T1	0.75840	0.85340	0.48130
T2	0.09095	0.06760	0.66070
T3	0.01051*	0.00032***	0.88680
T4	0.00047***	0.00414**	0.01254*
T5	0.46120	0.39300	0.88840
T6	0.78180	0.84210	0.79840
T7	0.03823*	0.37580	0.09324
T8	0.08108	-	0.07552
T9	0.92310	-	1
T10	0.82350	0.28180	-
*** $p < 0.001$ ** $p < 0.01$ * $p < 0.05$			

The prototype and Protégé have similar performance in completion time for most of the tasks. A significant difference ( $p < 0.05$ ) between the tools for T3 (examining siblings) was found, with Protégé taking less time than the prototype. Also, participants using the prototype took a notably longer time than those using Protégé for T4 (finding the path), with a high significance ( $p < 0.001$ ). For T7 (counting number of associations), the prototype considerably outperformed Protégé, with  $p < 0.05$ . As only the completion time for the correct answers was considered and most of the participants failed in the G3 tasks with Protégé, there are no completion time data for Protégé for T8 and T9 with the

small dataset, and for T10 with the large dataset. Therefore, the statistical test results are missing for these tasks.

**Participant Rating.** Participants rated the difficulty (lower is better) and confidence (higher is better) for each group of tasks on a five-point Likert scale ranging from 1 to 5. Figure 5.10a and Figure 5.10b show the percentage of participants' rating for both tools per task group, and Table 5.6 contains the rating *Wilcoxon test* results.

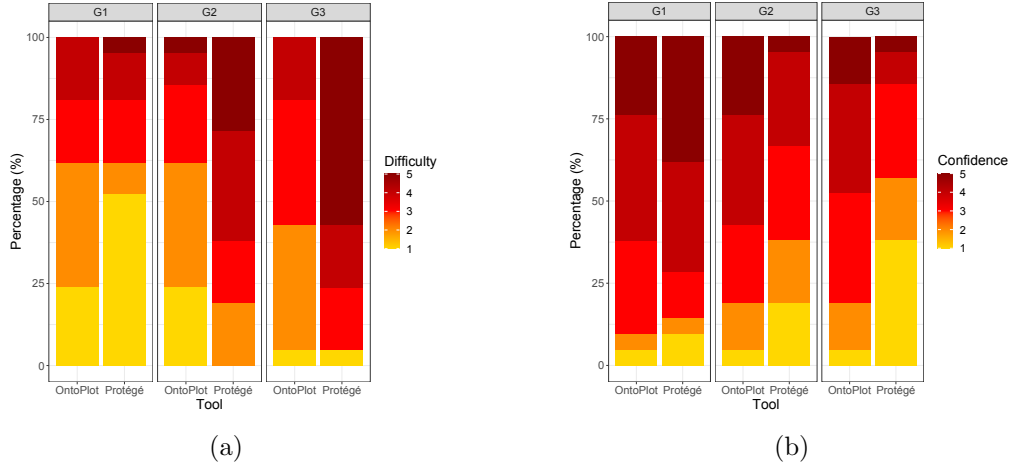


Figure 5.10: Participants' rating of the two tools: (a) difficulty rating for each group of tasks, (b) confidence rating for each group of tasks.

Table 5.4: Summary of statistical significance (Wilcoxon test) of differences in difficulty and confidence ratings, summarised in Figures 5.10a and 5.10b.

Group	Difficulty	Confidence
G1	0.36820	0.45550
G2	0.00071***	0.01865*
G3	0.00016***	0.00145**
***p < 0.001 **p < 0.01 *p < 0.05		

Overall, participants felt that performing G1 tasks in Protégé was slightly easier than in the prototype, and more participants rated Protégé difficulty 1 than for the prototype. From Figure 5.10a it is obvious that participants rated the G2 and G3 tasks in the prototype as far easier than in Protégé. The statistical test reveals high significances ( $p < 0.001$ ) between the prototype and Protégé for these two groups of tasks.

On the confidence of their answers, participants gave a higher rating to Protégé than to the prototype for G1 tasks, while the situation is reversed for G2 and G3 tasks. The *Wilcoxon test* shows the differences of confidence between the two tools for G2 and G3 tasks are significant ( $p < 0.05$  and  $p < 0.01$ ).

The result of the preference rating can be seen in Figure 5.11a. For G1 tasks, 52% of the participants preferred Protégé over the prototype, while for G2 and G3 tasks they mostly preferred the prototype with a high percentage of around 88%.

Figure 5.11b shows participants' rating of the learning effort for both tools, also on a five-point Likert scale ranging from 1 (easiest) to 5 (hardest). More participants rated

the learning effort 1 for Protégé than for the prototype, with 25% for Protégé and 15% for the prototype. On the other hand, 5% of participants rated the learning effort 5 for Protégé, while none gave this same rating for the prototype. Although the overall rating for the prototype is less than for Protégé, the *Wilcoxon test* shows there is no significant difference between the tools ( $p = 0.806$ ).



Figure 5.11: Participants’ rating of the two tools: (a) preference rating for each group of tasks, and (b) learning effort rating.

**Participant Feedback.** Some participants felt Protégé was more familiar and acceptable so it was easier to perceive structure, while others commented the prototype was easier to learn as it looked simple and clearly showed information. One participant commented that the prototype was powerful and its interactions were well-developed, but it needed users to follow the procedure to perform tasks, so it might be a challenge for a first-time user. There are also some contradictory results. A couple of participants rated the prototype better than Protégé on difficulty and confidence but at the end preferred Protégé over the prototype. One commented: “Protégé is more familiar and OntoPlot is too new... I need more time and effort to get used to OntoPlot”.

Aside from being unfamiliar with the prototype, participants commented the prototype was easy to use: “it is easier to collapse and expand in OntoPlot due to its seamless interaction area”, and “I quite like the mark label function as it kept reminding me previous targets”. They also commented that the prototype better supported G2 and G3 tasks: “I can directly see the answer from the colour”, “I found redrawing the visualisation by association and class is very helpful”, and “I prefer OntoPlot over Protégé as I felt could finish the task more quickly”.

Some participants liked the appearance of the prototype. They commented: “the colour in OntoPlot is nice”, and “I like the combination of list and visualisation”. Several participants felt it was hard to perceive siblings in the prototype, while others found the prototype helped in showing siblings: “the arrangement of siblings is efficient”. A few felt scrolling in the prototype is harder than in Protégé if the visualisation was large. This may suggest that users are more familiar with vertical scrolling than horizontal scrolling.

Participants also gave some suggestions on how to improve the prototype. One participant asked for a zooming in and out interaction such that users did not need to pan a lot when the visualisation was big. Some participants suggested adoption of useful functions

Table 5.5: Overall summary of statistical significance of results. The tool mentioned in the columns outperforms the other in terms of the given metric (O: OntoPlot, P: Protégé).

Group	Task	Accuracy	Time	Difficulty	Confidence	Preference	Learning Effort
G1	T1	-	O	O	P	P	O
	T2	P	P				
	T3	P *	P *				
	T4	P	P ***				
	T5	P	P				
G2	T6	O	P	O ***	O *	O	
	T7	O **	O *				
	T8	O ***	O				
G3	T9	O **	P	O ***	O **	O	
	T10	O ***	O				
***p < 0.001 **p < 0.01 *p < 0.05							

from Protégé to improve the prototype, such as being able to “search on object property” and “show all labels.” One participant suggested to “give an option to show all labels, or when clicking a class to redraw the visualisation should automatically label all associated (coloured) nodes”. One participant commented: “it is a bit confusing when collapse and expand ... probably can use animation to smooth transaction”.

**Summary.** Table 5.5 presents a summary of all the results. Overall, the results show that Protégé slightly outperformed the prototype for most of the hierarchy tasks (G1) on both accuracy and completion time. For G2 and G3 tasks, the prototype significantly outperformed Protégé on accuracy, but the completion time of both tools were similar. In general, participants’ difficulty rating, confidence rating and preference ranking were in line with the expectations of the study.

### 5.2.9 Discussion

The results show that Protégé slightly outperformed the prototype for G1 structure-related tasks, therefore rejecting the hypothesis H1. To explain this, it was observed that most participants made mistakes in the prototype on the tasks involving visual compression or cases where sibling classes were visually separated in multiple boxes (because some were grouped together as leaf nodes). Interestingly, although the indented list in Protégé is a common method for hierarchical visualisation, several participants mistakenly took the sibling above a class (at the same indentation level) as the parent of that class.

For G2 and G3 association-related tasks, the prototype outperformed Protégé on accuracy as expected (H2, H3). The main reason why participants got incorrect answers using Protégé was because they did not fully investigate the views to check all the associations listed in Protégé. Also, as Protégé cannot filter unneeded associations, some participants referred to the wrong associations when performing tasks. The reason for

errors in the prototype, however, was quite different. Most participants who did not get the correct answers forgot to click the classes specified in the tasks, so the prototype did not redraw the visualisation based on a particular class and they therefore referred to the wrong associations.

Surprisingly, the completion time of both tools for G2 and G3 tasks were similar. Observation revealed that most participants first spent some time thinking what they needed to do for each tool after reading the tasks. For Protégé, the process of browsing and distinguishing different associations was quite time-consuming. For the prototype, most participants spent time on interacting with the visualisation. As the prototype provides more interactions than Protégé, such as mouse hover, click, double-click, and shift-click, only a few users could clearly remember which interaction matched which function. As a result, most participants used multiple interactions in order to recall the functions during the experiment.

### 5.3 Prototype Refinement

The results and feedback from the prototype user study were considered, leading to some design and interaction refinements described in this section.

#### 5.3.1 Visual Representation

Some modifications were made to the visual representation for both hierarchy and association. The refined visualisation is shown in Figure 5.12. The details are discussed in the following sections.

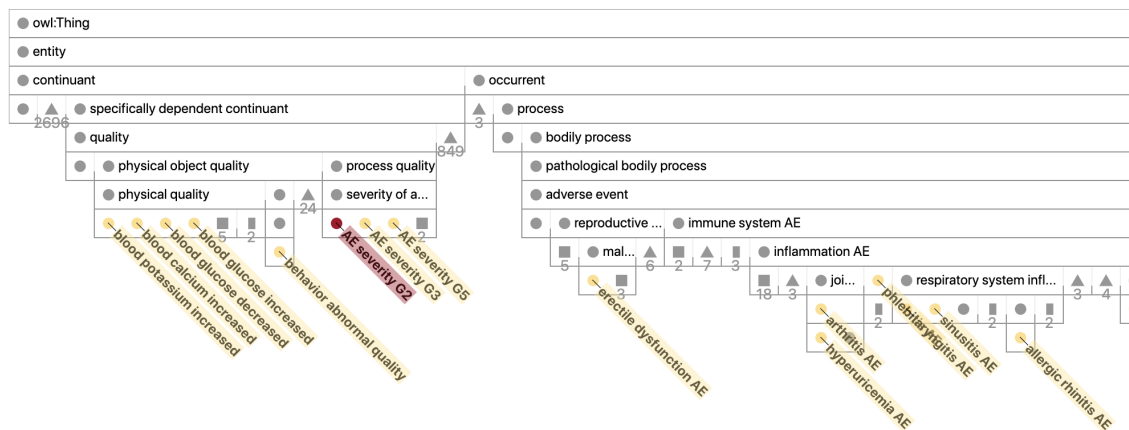


Figure 5.12: Overview of the refined visualisation.

##### 5.3.1.1 Different Separation of Boxes

In the prototype the leaf nodes are wrapped in a single box. During the user study this was found to be error-prone where sibling classes were separated in different boxes if some were leaf nodes but some had descendants.

To strive for consistent visual representation for the sibling classes and differentiate between neighbouring boxes containing siblings of the same parent class versus neighbouring boxes from different subtrees, the refined visualisation uses a partial and faint line in the first case and a solid line in the second case (see Figure 5.13).

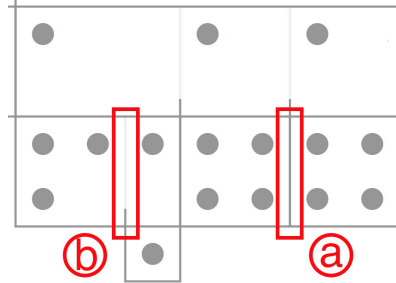
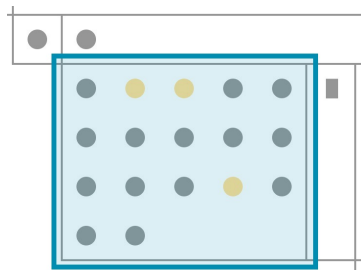


Figure 5.13: Solid lines are shown between different subtree boxes, e.g. ①. A partial and faint line is shown between sibling boxes, e.g. ②.

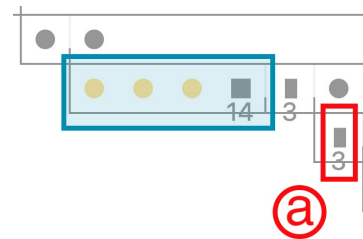
### 5.3.1.2 Compression of Leaf Nodes

There could be many leaf nodes in an ontology and the majority of them could be *uninteresting*. In the prototype, if a group of leaf nodes contains any leaf nodes that have associations, this group of nodes is not compressed (see Figure 5.14a).

To make the visualisation more compact and emphasise more of the *interesting* leaf nodes, in the refined visualisation the leaf nodes are ordered by the number of associations they have. The nodes that do not have any associations are compressed as a square (see Figure 5.14b). Note, the compression glyphs (square, thin block or triangle) are labelled with the number of hidden nodes inside them (for example, see Figure 5.14b ①).



(a) Compression of leaf nodes in the prototype.



(b) Compression of leaf nodes in the refined visualisation.

Figure 5.14: An example of refined compression of leaf nodes.

### 5.3.1.3 Class Labels

As discussed in Section 4.2.3.2, the prototype did not display class labels by default and they needed to be manually pinned to the visualisation for investigation. This gives little context of the classes in the ontology.

To address this and still avoid the visual occlusion, the labels of parent classes of displayed subtrees are shown greedily where space exists, as their boxes are often quite wide. If there is not enough space, the labels are truncated (see Figure 5.15).

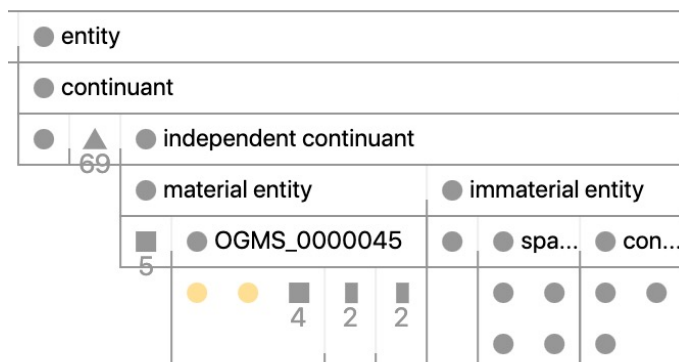


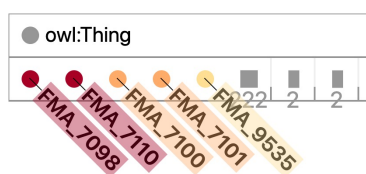
Figure 5.15: Parent classes are labelled where possible.

#### 5.3.1.4 Association Labels

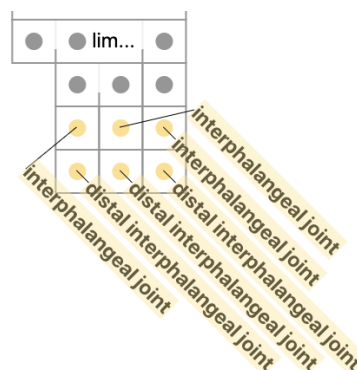
In the prototype, when a property is selected, the classes that have the associations with the selected property are highlighted by colouring their circle glyphs.

To emphasise the classes with associations in the ontology hierarchy, these classes are given association labels in the refined visualisation (see Figure 5.16a). The association labels are coloured the same as the circle glyphs, indicating the number of associations of the applied classes. Also, the association labels are positioned diagonally to allow labelling of neighbouring classes without occlusion.

As there are some cases where nodes with associations are within the hierarchy rather than just the leaf nodes (see Figure 5.16b), the visualisation allows clicking and dragging the association labels to arrange them.



(a) Classes with associations are labelled and coloured based on the number of associations.



(b) Association labels are initially positioned diagonally below the class they label to minimise overlaps, but can be manually dragged and arranged, if necessary.

Figure 5.16: Examples of association labels.



### 5.3.2 Interaction

There are some refinements to the way interaction works in the prototype, focusing on improving the usability to support the exploration of the ontology hierarchy and associations.

#### 5.3.2.1 Expanding and Collapsing

As discussed in Section 4.2.3.1, double-clicking on glyphs can either collapse or expand a subtree.

To help preserve users' mental map, the refined visualisation highlights the portion of the hierarchy being collapsed or expanded prior to the operation and then highlights the same portion for a short period (several seconds) after the operation has completed. This highlighting is shown in Figure 5.17. The specific highlighting period is calculated based on the size of the collapsed or expanded regions, giving enough time to observe the changes especially when a large subtree has been expanded.



(a) The highlighting before expanding the double-clicked triangle glyph. The box of the triangle is highlighted, indicating where the change will happen.

(b) The highlighting after expanding the double-clicked triangle glyph. The whole expanded subtree that was hidden by the triangle glyph is highlighted.

Figure 5.17: An example of highlighting portions when double-clicking a triangle glyph to expand a subtree.

Subtrees can be collapsed regardless of whether the classes they contain are interesting or uninteresting, or whether they include a selected class. If a subtree contains interesting classes, the glyph is displayed with a coloured glow, which indicates the maximum number of associations in the collapsed subtree (for example, see Figure 5.18 ①). If a subtree containing a selected class is collapsed, a pulsing red circle will be shown around the glyph for the collapsed subtree (see Figure 5.18 ②).

#### 5.3.2.2 Class Selection and Focus Mode

When a class is selected, it will be given a black outline in the prototype. In the refined visualisation, subtle arrows are drawn around the periphery, denoting the direction of the associations (one pointing in at the top-left if it is the target of the selected association type and another pointing out at the top-right if it is the source of the selected association type, see Figure 5.19 ①).

Additionally, a pop-up window is displayed below the selected node (see Figure 5.19 ②). This indicates the class label and the number of associations of the selected class.

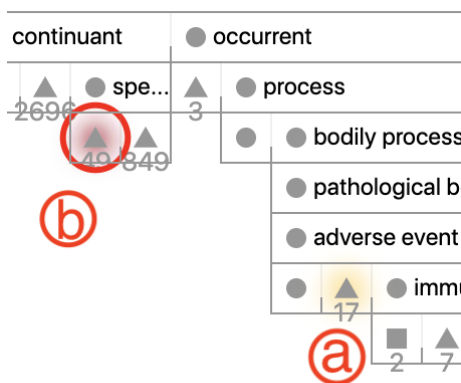


Figure 5.18: The collapsed subtree ① containing association classes is highlighted with a coloured glow. The collapsed subtree ② containing the selected class and association classes is highlighted with a pulsing red circle and a coloured glow. The red glow of ② and yellow glow of ① indicate the maximum number of associations inside the subtrees.

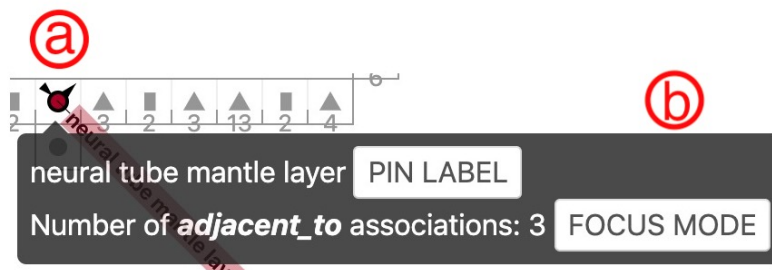


Figure 5.19: When a class is selected, OntoPlot highlights it ① and shows additional information and controls ②.

It provides a “PIN LABEL” button to mark the class with its label, rather than using shift-clicking as in the prototype where the key combination is not easy to remember.

In the prototype, selecting a class recomputes the *interesting* and *uninteresting* parts, which causes immediate visual change to the whole ontology hierarchy. In the refined visualisation, selecting classes does not cause any recomputing, but only updates the colour of the classes, which acts like the typical selection behaviour in most software. Instead, a *focus mode* is introduced and a “FOCUS MODE” button is provided in the pop-up window. Clicking the “FOCUS MODE” button after selecting a class will recompute the *interesting* and *uninteresting* parts and go into the *focus mode*, focusing on the associations for the selected class and compressing the *uninteresting* parts of the hierarchy (see Figure 5.20). While in this mode, a dark notification bar is shown at the top of the visualisation as a reminder (see Figure 5.20 ①). When in the *focus mode*, clicking the “FOCUS MODE” button in the pop-up will leave the mode.

In case the visualisation is scrolled away from the selected node, the refined visualisation shows a pulsing arrow at the boundary of the view that points to the glyph for the selected class. Clicking the arrow will scroll the visualisation to locate the selected class in the middle of the view.

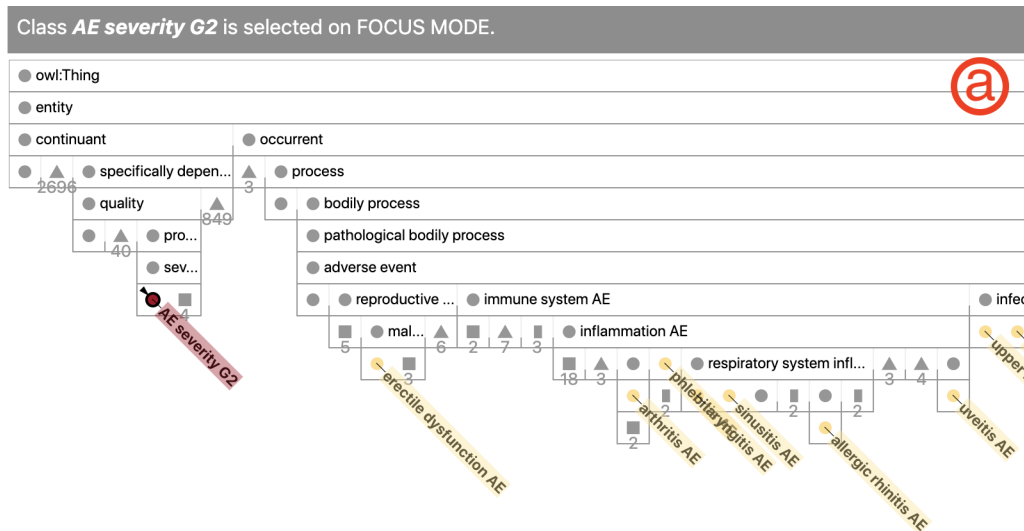


Figure 5.20: The *focus mode* for class AE severity G2 provides a compressed view that allows users to focus on an interesting subset of the ontology based on the associations for just this class. Note the notification bar at the top of the view.

### 5.3.2.3 Search

In the refined visualisation, the matching classes for the search term are not drawn with an additional circle around their class glyphs, as there could be many additional circles and this could cause a busy visualisation and make it hard to locate important classes.

Selecting a class from the search result list is equivalent to clicking a class in the refined visualisation, such that the visualisation is updated to centre on the selected class, colouring only the classes that have associations with the selected class.

## 5.4 Expert User Study

To test the usability of the refined visualisation of OntoPlot, an expert user study was conducted with 12 new participants, all domain experts or experienced ontology users.

### 5.4.1 Study Structure

The study design and tasks were the same as in the prototype user study described in Section 5.2. Regarding datasets, the prototype user study used a small manually constructed (and hence unrealistic) ontology for the training, and participants performed the tasks with ontologies CVDO and OCVDAAE. That study found little difference in the results between the two ontology sizes, hence the decision was made to evaluate only the larger ontology (OCVDAAE, 4,589 classes) in this expert study and use the smaller ontology (CVDO, 536 classes) for the training tasks. Therefore, a within-subjects design: 2 tools  $\times$  1 ontology size  $\times$  10 tasks (+ training) were used for this expert user experiment. The other procedure details were identical to the prototype user study.

### 5.4.2 Participants and Apparatus

All 12 participants had experience in the field of ontologies or knowledge graphs. Eleven of them identified as having experience using ontologies, including three with more than three years of experience. Ten participants had experiences using Protégé, one of whom had more than three years of experience. Another two participants had used other ontology tools, including the tools developed by the Gene Ontology Consortium and a proprietary tool used for a knowledge graph construction engine. Of the 12 participants, three were female and nine were male. Their age ranged from 18 to 41. All participants had normal or corrected-to-normal vision, and none suffered colour-blindness. None of them participated in the prototype user study.

The six participants recruited locally from Monash University used a 2.3 GHz Intel Core i5 laptop with 8GB of RAM, using a 24-inch monitor with a resolution of 3840x2160 pixels. The six participants recruited from other institutions did the experiment remotely, using their own computers at a resolution of 1600x900 pixels. For the remote participants, the experiments were observed via video call.

### 5.4.3 Results

All 12 participants completed the study. Again, accuracy and completion time was measured for each task. Difficulty level, confidence level, preference ranking and learning effort as rated by the participants were collected. The non-parametric *Wilcoxon test* was used to compare accuracy between the two tools, and the non-parametric *Whitney-Mann test* for unequal samples was used to analyse completion time data. For the rated results, the *Wilcoxon test* was also used.

**Accuracy.** Figure 5.21a shows the details of mean accuracy for each tool per task. Overall, participants achieved higher accuracy on most tasks with OntoPlot than with Protégé. The two exceptions are for T1 (finding parent) and T4 (finding path), which have equal accuracy (100%) for both tools. The *Wilcoxon test* revealed that for T8 (finding class with most associations), OntoPlot significantly outperformed Protégé ( $p < 0.05$ ).

**Completion Time.** Results for completion time are shown in Figure 5.21b. For most of the Hierarchy tasks (G1), participants spent less time on Protégé than on OntoPlot. Especially for T2 (finding children) and T3 (finding siblings), the *Whitney-Mann test* revealed that Protégé significantly outperformed OntoPlot ( $p < 0.01$ ). For T5 (finding common ancestor), OntoPlot and Protégé had very close completion time, with OntoPlot being slightly faster. Of the Association tasks (G2), for task T6 (finding individual associations) OntoPlot had a slightly longer average completion time than Protégé. The results show that tasks for finding and counting most associations (T7, T8), OntoPlot significantly outperformed Protégé ( $p < 0.001$ ). Highly significant differences were also found for the combined Hierarchy + Association tasks (G3) (T9, T10), with OntoPlot substantially outperforming Protégé. Taking accuracy into account, these results indicate that, especially for complex tasks (G3), OntoPlot requires substantially less time and achieves much higher accuracy than Protégé.

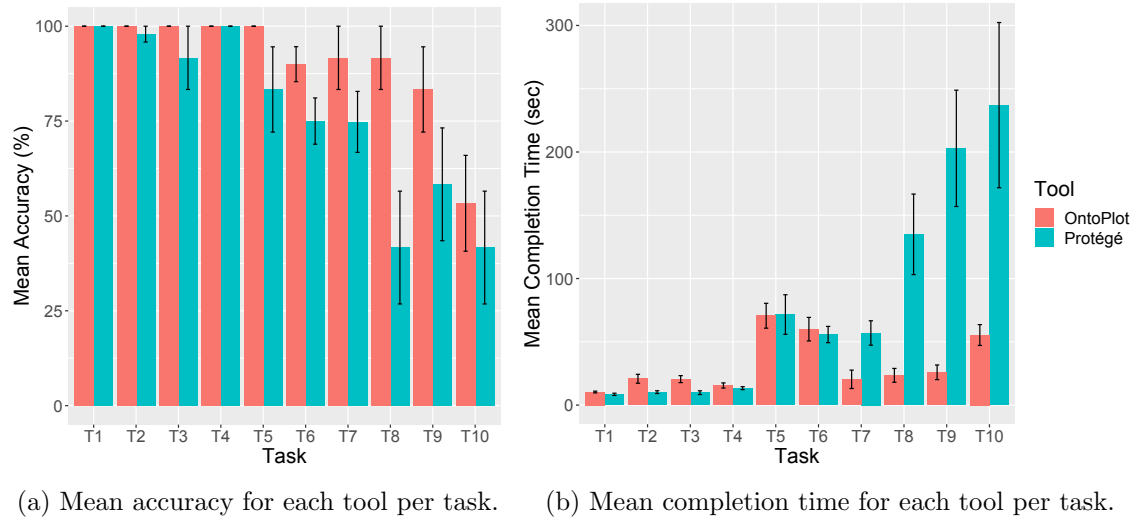


Figure 5.21: Participants' performance using the two tools in the expert user study.

**Participant Rating.** A five-point Likert scale ranging from 1 to 5 was again used to measure participants' rating of difficulty (lower is better) and confidence (higher is better) for each group of tasks and each tool. Figure 5.22a and Figure 5.22b show the percentage of participants' rating results, and Table 5.6 summarises the results.

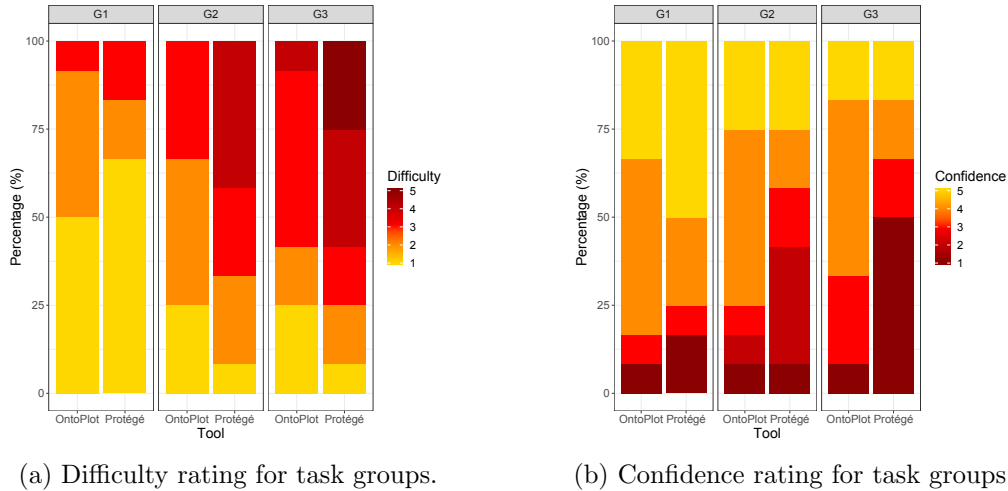


Figure 5.22: Participants' rating of the two tools in the expert user study.

Table 5.6: Summary of average difficulty and confidence ratings as shown in Figures 5.22a and 5.22b.

Group	Difficulty		Confidence	
	OntoPlot	Protégé	OntoPlot	Protégé
G1	1.583	1.5	4	3.917
G2	2.083	3	3.75	3.167
G3	2.417	3.5	3.667	2.5

Overall, participants rated G1 tasks performed in Protégé as slightly less difficult than in OntoPlot. For G2 and G3 tasks, participants rated OntoPlot as less difficult than Protégé. Three participants rated Protégé difficulty at 5 (highest) for G3 tasks.

When asked about confidence rating, participants felt slightly more confident with OntoPlot than with Protégé for G1 tasks and gave much higher confidence rating to OntoPlot for G2 and G3 tasks.

Figure 5.23a shows the result of the preference rating. For G1 tasks, seven participants preferred Protégé over OntoPlot, whereas the situation is entirely reversed for G2 and G3 tasks. All the participants preferred OntoPlot for these tasks.

The result of the learning effort rating is shown in Figure 5.23b, also using a five-point Likert scale ranging from 1 (easiest) to 5 (hardest). One participant rated learning effort 1 for OntoPlot, while one participant rated it 5 for Protégé. The average rating is 2.625 for OntoPlot and 3.25 for Protégé. There is no significant difference between the tools ( $p = 0.056$ ).

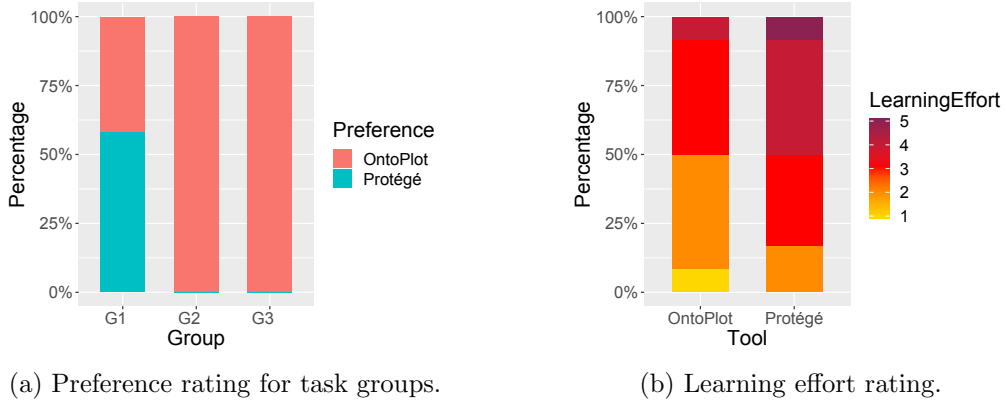


Figure 5.23: Participants' rating of the two tools in the expert user study.

**Participant Feedback.** At the end of the experiment each participant was given the chance to provide feedback and give comments. Some participants felt Protégé was more familiar and acceptable, e.g., commenting “The vertical aligned indented list is easier to perceive hierarchy structures”. Most of the participants gave positive feedback for OntoPlot, e.g., commenting “OntoPlot interface is more friendly”, “OntoPlot needs effort to learn, but makes tasks easier”, or “OntoPlot has more compact view of the ontology”. Some participants also provided more specific feedback such as “Lighter lines and darker lines are helpful for distinguishing siblings and non-siblings”, “The labels make finding associations much easier”, “Association labels are easy to read”, or “Tagging feature is nice”. One participant also commented on Protégé: “It is very difficult to find common ancestors with Protégé”.

A few participants also provided helpful feedback for further improvements of OntoPlot, e.g., “Probably can use colour coding for the sibling lines to make them more obvious”, “The subtle arrows could be more effective if they can indicate the number of pointing in and pointing out associations”, or “Probably can filter association classes further when there are many associations”.

Table 5.7: Summary of statistical significance for the expert user study results. The tool mentioned in the columns outperforms the other in terms of the given metric (O: OntoPlot, P: Protégé).

Group	Task	Accuracy	Time	Difficulty	Confidence	Preference	Learning Effort
G1	T1	-	P	P	O	P	O
	T2	O	P**				
	T3	O	P**				
	T4	-	P				
	T5	O	O				
G2	T6	O	P	O	O	O	
	T7	O	O***				
	T8	O*	O***				
G3	T9	O	O***	O	O	O	
	T10	O	O**				
Significance: ***p < 0.001    **p < 0.01    *p < 0.05							

**Summary.** Table 5.7 presents a summary of all the results. Overall, OntoPlot moderately outperformed Protégé on accuracy for most tasks, and significantly (i.e., statistically significantly) outperformed Protégé for the task T8. On completion time, Protégé outperformed OntoPlot for most G1 tasks and significantly outperformed it on two tasks, while OntoPlot significantly outperformed Protégé for most G2 and G3 tasks. No significant difference was revealed by the statistical test for the participants’ rating data. These results are consistent with those from the prototype user study, while in the expert study the users had noticeably better accuracy rates using both tools and they performed significantly faster using OntoPlot than Protégé on the G2 and G3 (association) tasks.

#### 5.4.4 Discussion

The expert user study shows that OntoPlot did not perform significantly differently than Protégé for Hierarchy tasks (G1) on accuracy, which confirmed the hypothesis H1 (Section 5.2.4). A common error made by several participants in Protégé was to mistake the sibling shown above a class (at the same indentation level) as the parent of that class, often when there was a distance between them in the indented list, which was the same mistake found in the prototype user study but occurred less. On completion time, Protégé significantly outperformed OntoPlot for the tasks involving finding children and siblings. This can be explained by the fact that most participants were Protégé users and were familiar with the indented list for showing hierarchy structure. Also, in order to test the participants’ perception of glyph compression, this group of tasks was designed to force participants to collapse or expand the subtrees. The participants spent some time on understanding which glyph or class they should collapse or expand in OntoPlot, and double-checked their answers. In Protégé most of the participants were able to skilfully

interact with the indented list. However, for the finding common ancestor task, the participants spent a little less time in OntoPlot than in Protégé as they could mark the classes by labels, and this made the task easier.

For association-related tasks (G2 and G3), OntoPlot outperformed Protégé on most of the tasks as expected (accepting H2, H3). Especially, for the completion time, there are some significant differences between the tools. The main reason why participants spent more time in Protégé was because in Protégé a user cannot select both classes and associations at the same time. Thus, the participants had to distinguish different classes or associations by themselves. The reason why OntoPlot took marginally more time for task T6 (finding individual association classes) was that some participants spent some time on scrolling the visualisation or dragging the association labels.

## 5.5 Visualisation Extensions

This section describes a number of extensions to the visualisation presented in Section 5.3, in order to better support large ontology hierarchies and associations. Some extensions are based on the results of the expert user study, focusing on the usability of the visualisation.

### 5.5.1 Hill Glyph

As introduced in Section 4.2.2.1, there are three structures that can be compressed in the visualisation. In addition to these three structures, a fourth case was identified to further compress the *uninteresting* ontology sections to give a more compact visualisation emphasising the *interesting* parts.

When more than one of the previously described cases occur as siblings, i.e., multiple compression glyphs happen to be siblings (highlighted with red background in Figure 5.24a), these glyphs are replaced with a single hill glyph (highlighted with red background in Figure 5.24b), labelled with the total number of nodes hidden inside this glyph. To allow this replacement, the classes are firstly clustered by whether there is any association with them or within the subtrees they are the root of, then they are ordered alphabetically. The glyphs of classes and subtrees that do not have any associations are arranged together and then replaced by a hill glyph. The same as in the previous design, the leaf nodes are ordered by the number of associations they have. The leaf nodes without any associations are grouped and replaced by a square (see Figure 5.14b). In order to put this kind of square together with other compression glyphs, the squares now are partitioned out from the leaf boxes containing associations (see Figure 5.24a ①) into separate boxes (see Figure 5.24a ②).

As with previous described compression glyphs, double-clicking can expand hill glyphs. As hill glyphs apply two levels of compression (compressing compression glyphs), in order to allow collapsing the expanded sibling glyphs back to a hill glyph, after expanding a hill glyph, a “hill collapse button” with two indicator lines is drawn above these sibling glyphs on their parent box (see Figure 5.24a ③). Users can double-click this “button”



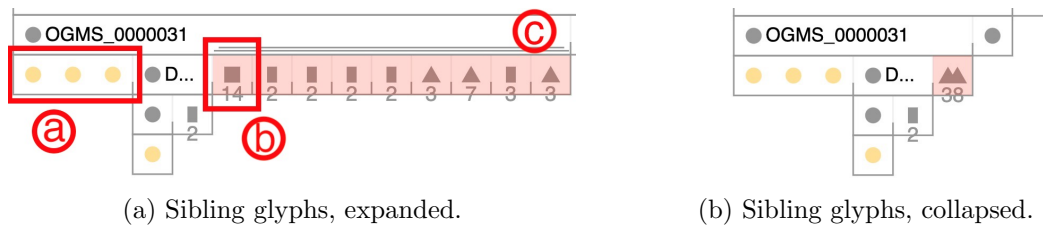


Figure 5.24: An example of hill glyph visual compression.

to collapse the sibling glyphs to a hill glyph. Double-clicking any of the collapsed sibling glyphs expands that subtree, as before.

### 5.5.2 Hidden Structure Visual Summary

To assist users to better understand the structure hidden by compression glyphs, when users hover on a glyph with mouse cursor, a miniature representation of the compressed structure will be displayed in a pop-up window (see Figure 5.25), giving information about the shape and the size of the compressed part. This allows the hidden structure to be quickly observed without expansion.

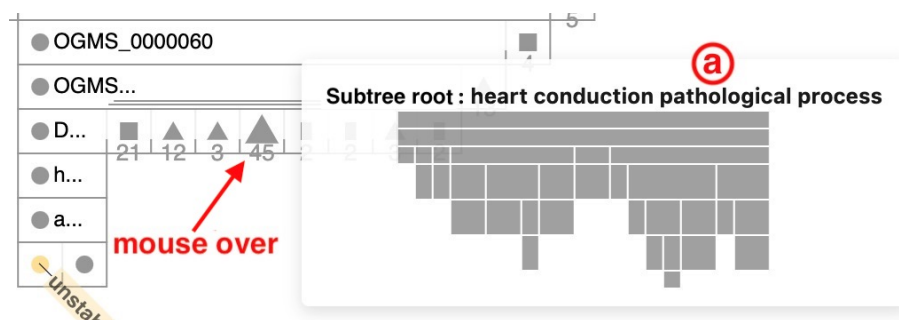


Figure 5.25: Visual summary of the hidden structure in a triangle glyph.

If the compressed subtree being examined contains a subtree root class, which is a single class at the top level, such as the glyph presenting a chain or a subtree, the label of the subtree root class will be shown on the top of the miniature in the pop-up window (for example, Figure 5.25 (a)), facilitating easy browsing of the contextual information through the ontology.

### 5.5.3 Scrollable Class Labels

As mentioned in Section 5.3.1.3, class labels are greedily displayed in the boxes beside their class circle glyphs. As ontologies can be very wide and the parent boxes for subtrees are also often wide, labels displayed at the left-hand side of a box may have been scrolled off the screen.

To address this and preserve the context for the classes in user viewpoint, the label positions are automatically adjusted during scrolling to be positioned at the leftmost on screen within their containing boxes, where the label still fits within the box, as shown in Figure 5.26.

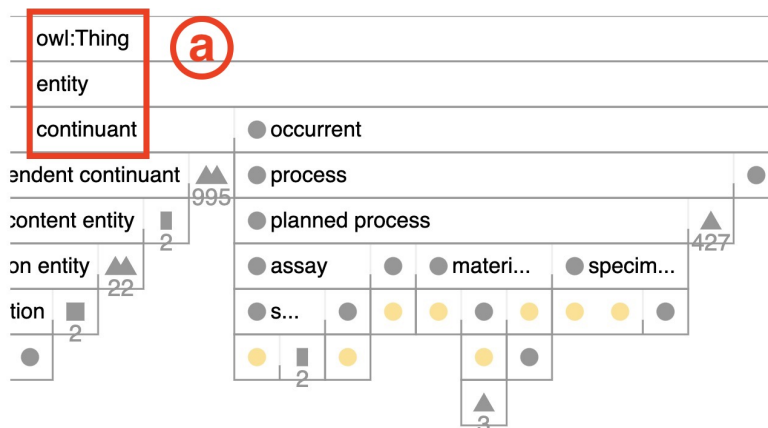
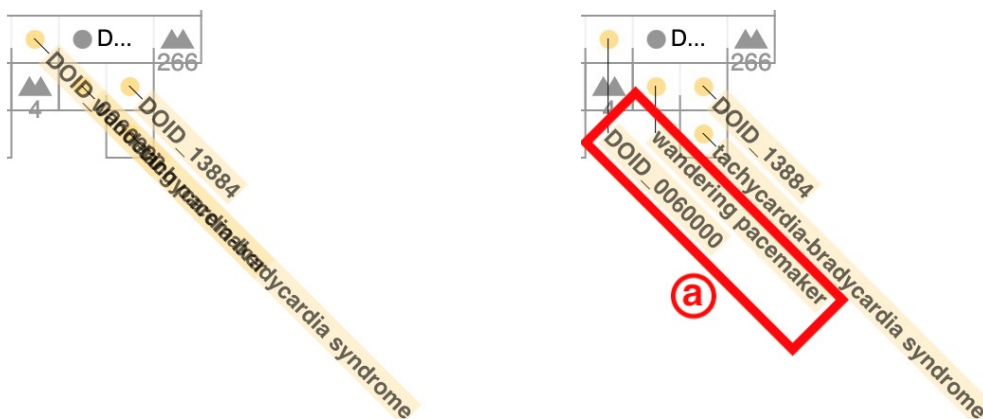


Figure 5.26: A situation when ancestor class circles are scrolled out of the view. ① the scrolled labels are automatically repositioned during scrolling to stay visible on screen.

#### 5.5.4 Non-overlapping Association Labels

As described in Section 6.1.1.2, to reduce the occlusion of neighbouring association labels, the labels are positioned diagonally. Classes with associations can occur densely and some association labels can overlap, especially when there are nodes with associations within the hierarchy rather than just the leaf nodes. Although the visualisation allows users to manually drag and arrange the association labels, a new technique is added into the visualisation to automatically shift the overlapping association labels vertically down where space exists (this space is generally free due to ontologies typically being wide and flat). Figure 5.27 shows an example.



(a) Overlapping association labels.

(b) Previously overlapping association labels ① are automatically shifted vertically down to eliminate overlaps.

Figure 5.27: An example of automatic shifting of overlapping association labels in (a) to get non-overlapping association labels in (b).

### 5.5.5 Minimap

This section describes a new technique, the minimap, shown at the bottom of the main visualisation, which gives an overview of the whole ontology hierarchy and associations and provides a navigation through the ontology.

This minimap uses the full width of the screen and shows the whole hierarchy scaled to fill this width. The parts of the ontology that are compressed are shown as lighter grey (for example, Figure 5.28 (a)) and the other parts are shown as darker grey (for example, Figure 5.28 (b)). The association classes are highlighted using coloured circles on the minimap, matching the colours in the main visualisation (for example, Figure 5.28 (c)). These circles are shown with a larger size, independent to the size of the non-association classes in the minimap that might be too small to see. The colours of the hierarchy and circles are synchronised with the main visualisation while users interact with it, such as selecting, collapsing or expanding.

If the entire ontology is not visible in the main visualisation, an *overlay* will be drawn on top of the minimap to indicate the visible portion of the ontology (see Figure 5.28 (d)). When users scroll the main visualisation, the overlay on the minimap will be updated accordingly. As some sections of the hierarchy will usually be compressed (such as the middle of Figure 5.28) and the minimap shows the entire uncompressed hierarchy, the width of the overlay region can change during scrolling.

If users click on the minimap, the main visualisation will be scrolled and centred on the location of the class that the user clicked.

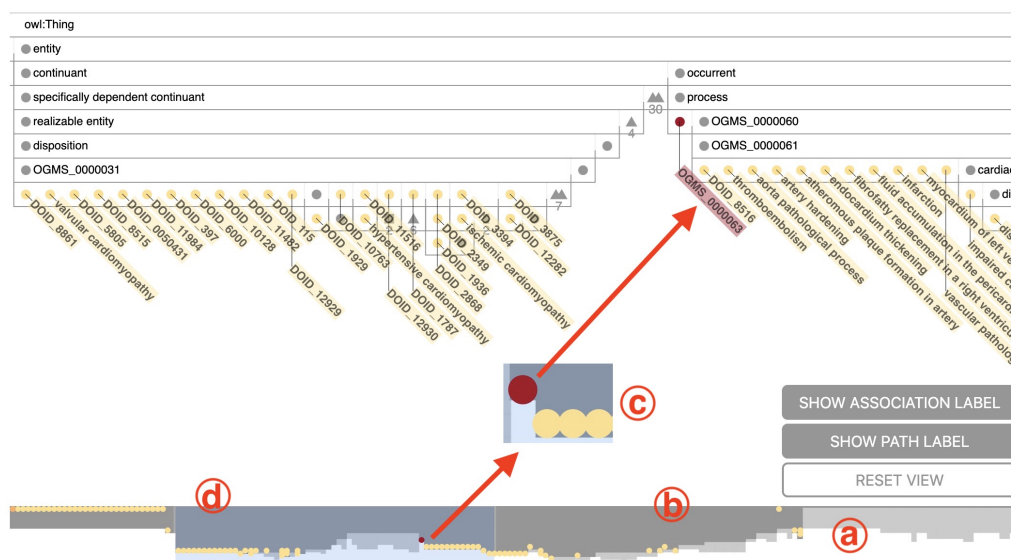


Figure 5.28: The minimap of the main visualisation is displayed at the bottom.

### 5.5.6 Import and Export

To make OntoPlot more responsive to general audiences, file import and export functionalities were added.

Users can upload and visualise their own ontologies. The supported input format is Web Ontology Language (OWL). When an ontology is uploaded by a user, it is appended to the top of the ontology list on the left pane, allowing later use by the user. The uploaded ontology is then preprocessed and visualised. As OntoPlot is a web-based software and implemented with JavaScript, the OWL files need to be preprocessed and transformed into CSV (Comma-Separated Values) files that are required by OntoPlot. Thus, a JAR (Java ARchive) file was implemented using the Java-based OWL API (Horridge and Bechhofer, 2011) to convert the ontology OWL files into CSV files. The JAR file is stored on the server side and this transformation is performed on the server when a user first uploads an OWL file.

To allow users to present the visualisations of ontologies and associations easily in their publications, OntoPlot supports export of the current visualisation in Scalable Vector Graphics (SVG) format. Users can prepare a compressed visualisation showing information of interest to make a figure. The label pinning allow users to show selected labels and produce a figure that highlights specific classes. This exported figure shows the whole visualisation not just the view (which may show only a section of the ontology).

## 5.6 Performance Measure

As OntoPlot employs visual compression techniques to collapse the parts of an ontology that are not interesting in terms of the selected property, the loading and processing time for OntoPlot to visualise an ontology cannot be measured solely based on the size of the ontology and the number of the associations that the ontology contains. The structure of an ontology, and more specifically, the distribution of the ontology associations in the hierarchy, also affects the performance of OntoPlot.

Some rough measurements have been done to explore the performance of OntoPlot when initially visualising certain ontologies of different sizes and structures, with different properties selected. The selection of properties aimed to cover different resulting numbers of compressed branches and uncompressed classes. The results are summarised in Table 5.8. Note, the number of classes of each ontology includes duplication for multiple inheritance. The tests were done on the local host run on a laptop with a 2.3 GHz Intel Core i5 CPU, 8GB of RAM and Intel Iris Plus Graphics 640 GPU.

After an ontology has been initially visualised, OntoPlot does caching of any compressed and uncompressed parts of the ontology to avoid recomputing of the whole visualisation during user interactions and view changes, and keeps on caching the changed parts of the ontology.

## 5.7 Conclusion

This chapter demonstrated the process of visualising homogeneous associations in ontology hierarchies. It includes prototype design, prototype evaluation, prototype refinement, expert user study and extension works. The results of the expert evaluation, along with the results of the prototype evaluation, demonstrate the value of OntoPlot for analysis

Table 5.8: Summary of OntoPlot performance time.

Ontology	No. of Classes	No. of Associations	No. of Compressed Branches	No. of Uncompressed Classes	Time
OCVDAE	4,589	8,308	55	159	1.40s
			65	638	3.49s
			30	1,347	8.73s
ODNAE	8,184	810	13	23	2.48s
			7	874	4.61s
			8	5,590	19.83s
CIDO	44,024	3,950	13	56	4.41s
			220	14,567	42.28s
			342	15,525	141.32s
GO	758,689	19,538	29	31	184.23s
			118	933	253.31s
			2,467	212,536	1012.74s

tasks as well as showing a noticeable improvement was achieved in the refined OntoPlot, especially for the hierarchy-based tasks.

The next chapter will describe the approaches for visualising heterogeneous associations that involve more than one property.



## Chapter 6

# Visualising Heterogeneous Associations

Chapters 4 and 5 presented the work for visualising large ontologies and their homogeneous associations, which address the research questions one and two. This chapter answers the third research question (see Chapter 1): how can heterogeneous associations (involving multiple properties) be effectively visualised in an ontology?

This chapter is organised as follows. Section 6.1 explores the design space and describes the approaches for showing multiple properties in the ontology hierarchy. More specifically, the new interactive visualisation techniques described in this chapter support multi-property selection and multi-class selection. These techniques are implemented as extensions to OntoPlot. Relevant terminologies for the techniques are also introduced in this section. Section 6.2 presents two case studies, illustrating the common tasks involving heterogeneous associations that domain experts undertake with ontologies, and demonstrating how the new techniques in OntoPlot support these tasks via a cognitive walk-through. As the formatting of this section tries to keep the figures and associated text close together to show step-by-step processes during the walk-through, there is some empty space left on some pages.

### 6.1 Visual Design

This section firstly describes the design for visualising multiple properties. Then it describes the approach for the selection of multiple classes.

#### 6.1.1 Visualising Multiple Properties

As mentioned in Section 5.1.1, when an ontology is loaded, the object properties found in that ontology are listed on the left-hand side of the OntoPlot interface. OntoPlot employs a new approach to allow users to simultaneously select more than one property that they are interested in.

To indicate the matches between properties and classes (U10, see Table 2.5), when users hover their mouse over a selected property, the circles of the classes that have that

property will be enlarged in the main visualisation and in the minimap. When users hover over a class, the properties applied to that class are highlighted and other properties become more transparent.

### 6.1.1.1 Pie Glyphs Attempt

The first attempt for visualising multiple properties in the ontology hierarchy used pie glyphs.

When users select more than one property, the circle glyphs are replaced with pie glyphs. The pie glyphs are partitioned with each sector responsible for one selected property. The clockwise order of the sectors matches the order of the selected properties on the property list. The colour of a sector uses the same colour scale introduced in Section 5.1.1.1, indicating the number of associations linked by the property that the sector represents.

For example, as shown in Figure 6.1, when two properties *BFO\_0000066* and *BFO\_0000082* are selected, the pie glyph is divided into two sectors. Following the clockwise direction, sector 1 is responsible for the first property *BFO\_0000066* and sector 2 is for the second property *BFO\_0000082*. To give more horizontal space for the visualisation, all the figures in this chapter show a cropped property list, superimposed on the main visualisation rather than how it is displayed in a left-hand pane in OntoPlot.

If only one property is selected, the behaviour would be as described in the previous chapter.

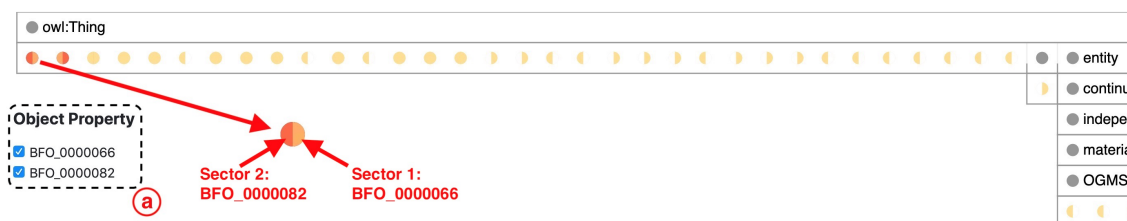


Figure 6.1: The visualisation with two properties selected. ① is the cropped property list showing property *BFO\_0000066* and *BFO\_0000082* are selected.

Figure 6.2 shows the situation for three properties. When the third property *BFO\_0000167* is selected, its corresponding sector 3 is added into the pie glyph as a third segment.

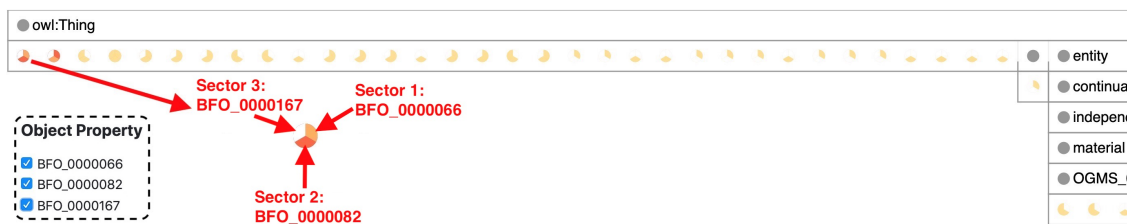


Figure 6.2: The visualisation with three properties selected. Note, as there is no association linked by the third property *BFO\_0000167* to the class shown in the callout, sector 3 of that class is coloured white.



The main drawback of using pie glyphs to visualise multiple properties is that when there are many properties selected, sectors will become too small to distinguish. Also, associating each property to its corresponding sector requires mental effort since they each have different orientations.

### 6.1.1.2 Association Labels

To address the problems of using pie glyphs, an alternative solution based on the association labels was devised.

As mentioned in Section 6.1.1.2, when a property is selected, the classes that have associations linked by this property are all highlighted by colouring the background of their association labels.

For visualising multiple properties, a series of small boxes are prepended to the association labels, where a box is responsible for a property and the order of boxes matches the order of the selected properties. The colour of each box reflects the number of associations linked to the labelled class by the property that the box represents, using the same colour scale as in Section 5.1.1.1 (U10, see Table 2.5). Figure 6.3 shows an example.

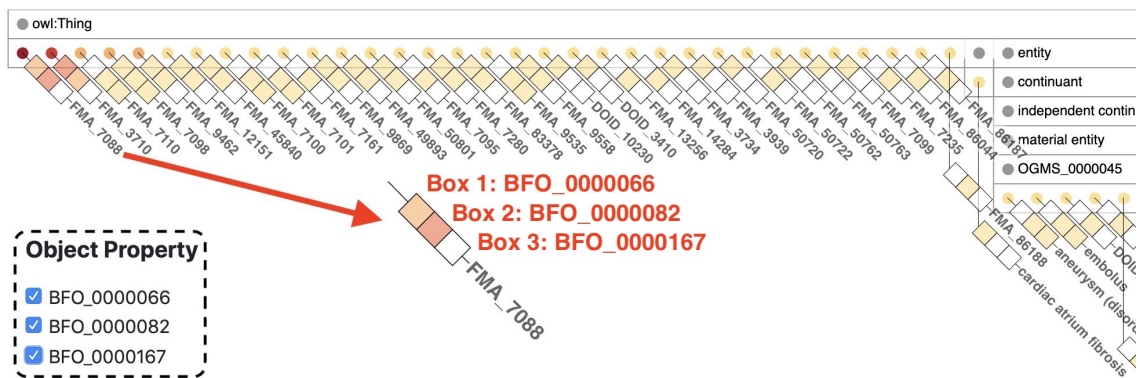


Figure 6.3: The association labels when three properties are selected. Box 1 is responsible for the first property *BFO\_0000066*, and so on.

Some interactions are employed to facilitate the matching between boxes and properties (U10, see Table 2.5). When a user hovers over a box, its corresponding property is highlighted by a thicker border and other properties on the property list are faded (see Figure 6.4 (a)). All the boxes on the association labels across different classes that are responsible for the same property are also highlighted with thicker borders (see Figure 6.4 (b)). A tooltip is shown, giving the number of associations for the class with the corresponding property (see Figure 6.4 (c)).

To allow users to quickly investigate the association classes for each class linked by each selected property (U12, see Table 2.5), when users double-click a box on the association label of an interesting class to select that box, all the other association labels will be hidden. Only the classes linked by the corresponding property of the selected box to the class of interest are highlighted with coloured labels (for example, see Figure 6.5). As the associations between classes are many-to-many relationships, this approach facilitates the exploration of associations for individual classes without losing current selection of classes.



Figure 6.4: An example of the visualisation when a user mouse hover over a box of an association label.

Users can double-click the selected box to deselect it. Then the visualisation leaves this mode and shows all the hidden association labels.

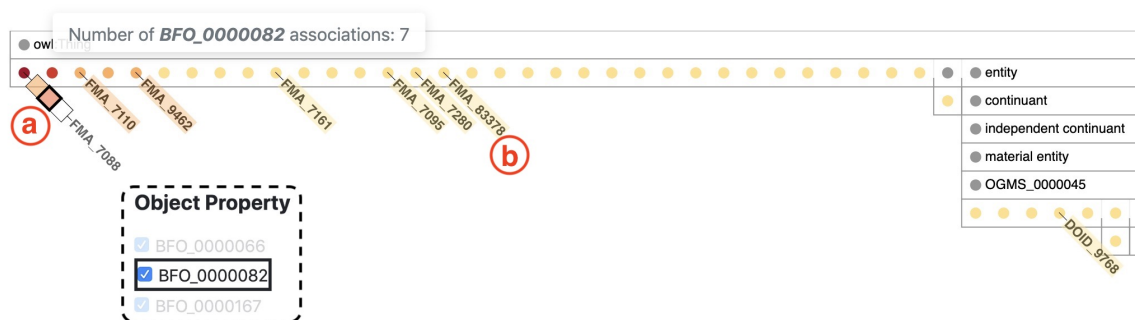


Figure 6.5: When a user double-clicks the second box ① on the association label of the FMA\_7088 class, only the classes having *BFO\_0000082* association with FMA\_7088 are labelled (for example, ②).

### 6.1.1.3 Union and Intersection

There are two operations for multiple property selection: *union* and *intersection*. For *union*, the classes that have *any* of the selected properties are highlighted (coloured and labelled, see Figure 6.6a). For *intersection*, only the classes that have *all* the selected properties are highlighted as shown in Figure 6.6b (U11, see Table 2.5).

There is a special case for *intersection*. In the *intersection* mode, if a class has all the selected properties but associated with different classes, this class will be coloured as grey and the boxes on its association label are all white (see Figure 6.6b ①). The classes linked by all selected properties to another class are coloured and labelled with coloured boxes (see Figure 6.6b ② and ③).

When multiple properties are selected, the system automatically checks all the classes that the selected properties are applied to. The *intersection* option is only available if there is at least one class that has all those selected properties applied to it.

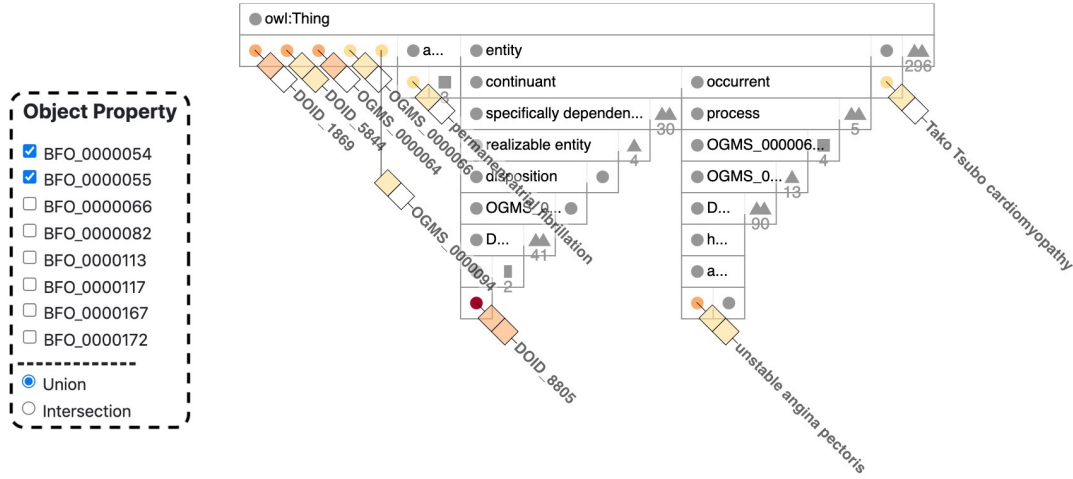
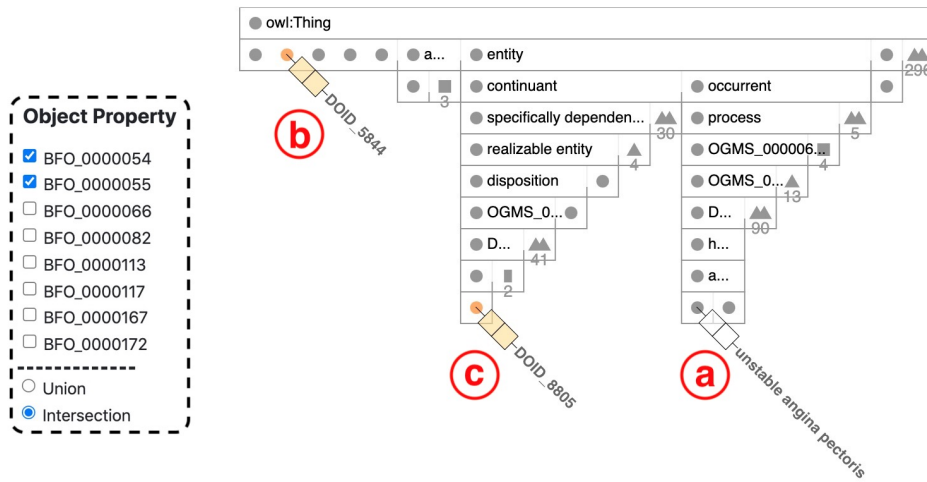
(a) Union of *BFO\_0000054* and *BFO\_0000055* properties.(b) Intersection of *BFO\_0000054* and *BFO\_0000055* properties. The classes marked as (b) and (c) are linked by both properties to each other. While the class marked as (a) has both properties applied to it, it is linked to two other classes (not shown) by these two properties respectively.

Figure 6.6: Union and intersection for multiple properties.

### 6.1.2 Multiple Class Selection

Another new approach in OntoPlot is allowing users to select multiple classes to perform analysis (U12, U13, see Table 2.5). Users select more than one class by shift-clicking classes (holding the shift key and clicking multiple classes). The selected classes are given thick blue borders to indicate the selection. The new approaches supporting multiple classes selection are described in detail as follows.

#### 6.1.2.1 Highlighting Property List

When users select a class, the property list is reordered and the properties that are applied to the selected class will be moved to the top of the list and given a grey background colour.

If multiple classes are selected, the properties that are applied to all the selected classes will be given a dark grey background colour (see Figure 6.7).

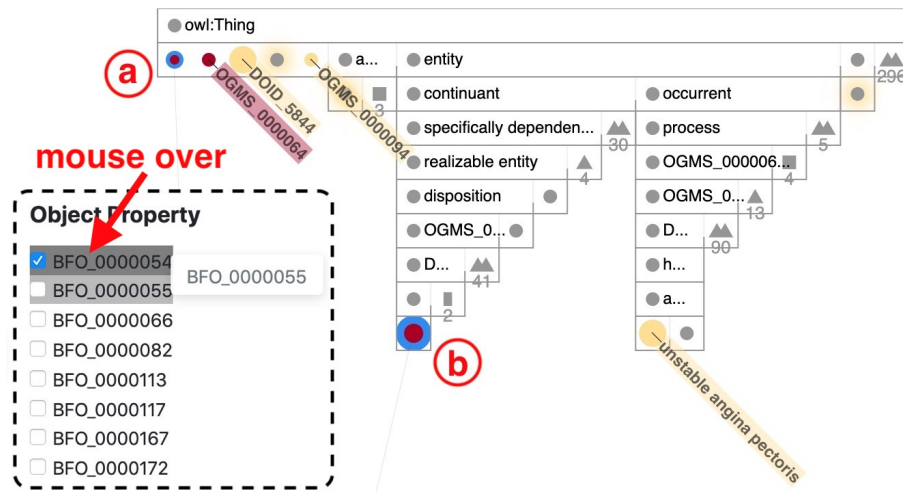


Figure 6.7: When the classes marked as (a) and (b) are selected, the property list is updated. Both of the *BFO\_0000054* and *BFO\_0000055* properties are moved to the top of the list, with *BFO\_0000054* having a dark grey background, indicating *BFO\_0000054* is applied to both selected classes. When a user hovers over the *BFO\_0000055* property on the property list, the class (b) is enlarged, indicating *BFO\_0000055* is only applied to class (b).

### 6.1.2.2 Colour Glow

As discussed in Section 5.1.2.3, when users select a class, the visualisation is updated and only the classes that have associations with the selected class are labelled and coloured.

As the new system allows selection of multiple classes, in order to let users be aware of all the classes that have associations, the classes that have associations but not with the selected class are turned grey, surrounded in a coloured glow matching their original colours (see Figure 6.8). This gives a hint to support users to inspect which classes they might want to further select to perform analysis.

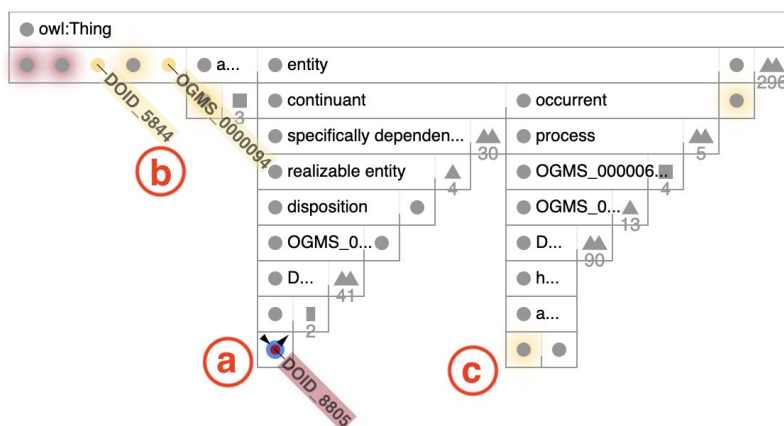


Figure 6.8: When selecting the class marked as (a), the classes that have associations with it are labelled and coloured (for example, (b)). The classes that have associations with other classes rather than (a) are surrounded by a coloured glow (for example, (c)).

### 6.1.2.3 Grid Labels

When selecting multiple classes, a grid label that consists of a grid and label text is prepended to each selected class as shown for the two selected classes in Figure 6.9 (a) and (b) (U12, U13, see Table 2.5), instead of its normal label.

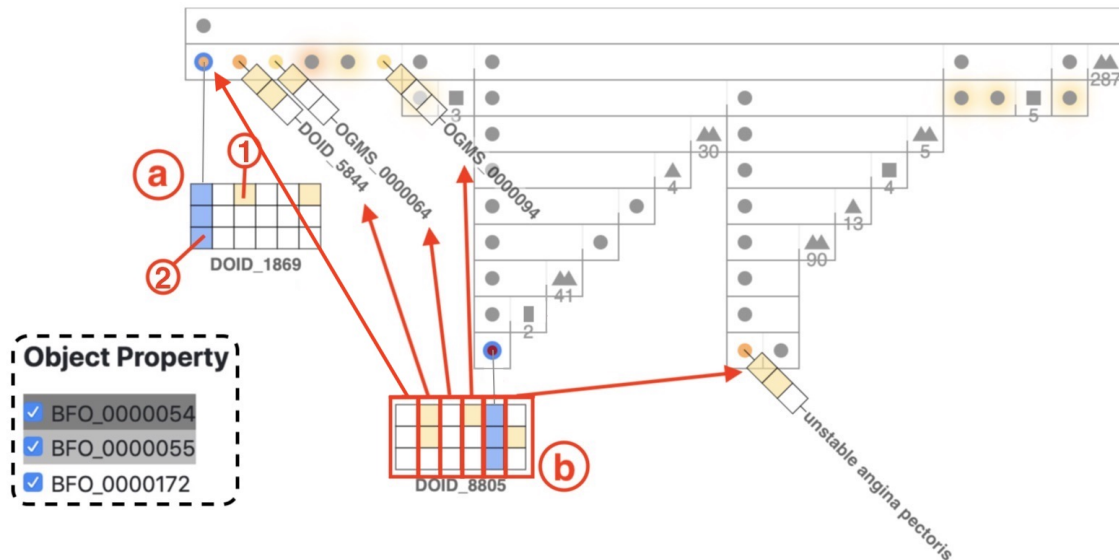


Figure 6.9: Grid labels of the selected classes DOID\_1869 and DOID\_8805, with three properties *BFO\_0000054*, *BFO\_0000055* and *BFO\_0000172* selected.

In each grid, the rows represent selected properties and the columns represent classes. The classes here are the association classes of any selected class and the selected classes themselves. The order of the rows again matches the order of the selected properties and association label boxes described in Section 6.1.1.2. The order of the columns matches the left-to-right position of the association classes and the selected classes in the main hierarchy visualisation (see highlighting of Figure 6.9 (b)).

If two association/selected classes happen to be ancestor-descendants, the ancestor class column is always put to the left of the descendant class column. If an association class happens to have identical classes (caused by multiple inheritance), only one column is created in the grid to present all those identical classes.

A leader line of each grid label is drawn between the selected class circle in the hierarchy and the column representing the selected class in the grid label. If users accidentally select classes that are identical but under different parents in the hierarchy, only one grid label will be generated, and the grid label will be linked to all the selected identical classes via additional leader lines.

To reduce the visual occlusion, the leader lines are lighter than the other visual elements (to show them clearly, they have been manually darkened for some figures in this chapter). Only when users mouse over a grid label, the leader line of that label is darkened.

Associating both a column and a row, a cell in the grid is responsible for a class and a property. The cell that belongs to an association class is coloured based on the colour scale introduced in Section 5.1.1.1, indicating the number of associations linked by the property that the cell stands for, between the association class and the selected class (for example,

Figure 6.9 ①). The cells positioned on the column representing the selected class are all coloured as blue (for example, Figure 6.9 ②).

As users might select numerous classes, in order to keep the grids organised for easy comparison, the grids are arranged greedily in vertical lines starting from the left, and their order matches the left-to-right positions of the selected classes (see Figure 6.10).

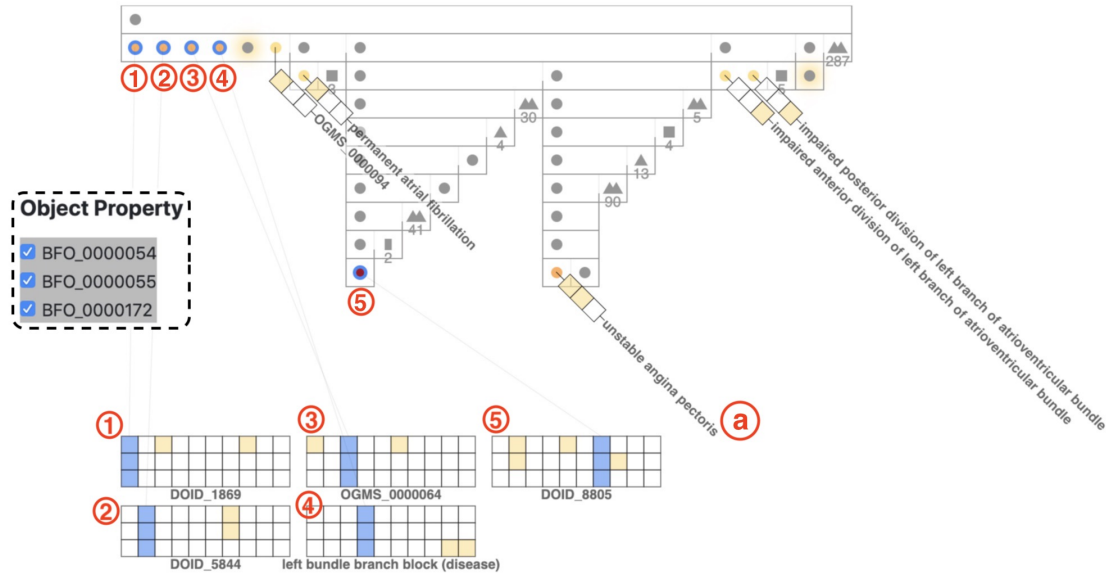


Figure 6.10: Five classes are selected in the visualisation and five grid labels are generated. The numbers ①②③④⑤ labelling the grid labels match the numbers labelling the selected classes, indicating the corresponding grid label and class respectively. Note, leader lines are shown for each grid label, though very faintly.

To avoid overlaps between the main visualisation and the grids, the grids are initially positioned below the longest association label (for example, Figure 6.10 ①). It is also possible for users to drag and arrange the grid labels manually.

As the main visualisation can be wider than the view, when users scroll the view, the grid labels will be kept on the view. If an association class is located outside of the current view, users can double-click a cell on the grid label and the visualisation will be centred on the association class that the cell represents. If the association class has identical classes in the hierarchy due to multiple inheritance, when users double-click the same cell repeatedly, the visualisation will scroll to cycle between each of the identical classes in turn.

When users brush over a cell, all the columns in all the grid labels representing the same class will be highlighted by thick black borders (for example, Figure 6.11 ①), and all the circles in the hierarchy that represent the same class will be enlarged and highlighted by thick black borders (see Figure 6.11 ②). This allows an easy comparison of associations between classes. The corresponding circles in the minimap are also enlarged simultaneously. The property that the hovered over cell represents is also highlighted in the list (see Figure 6.11 ③). Additionally, the label of the hovered over class will be shown on the pop-up tooltip on the upper right of the grid (see Figure 6.11 ④).



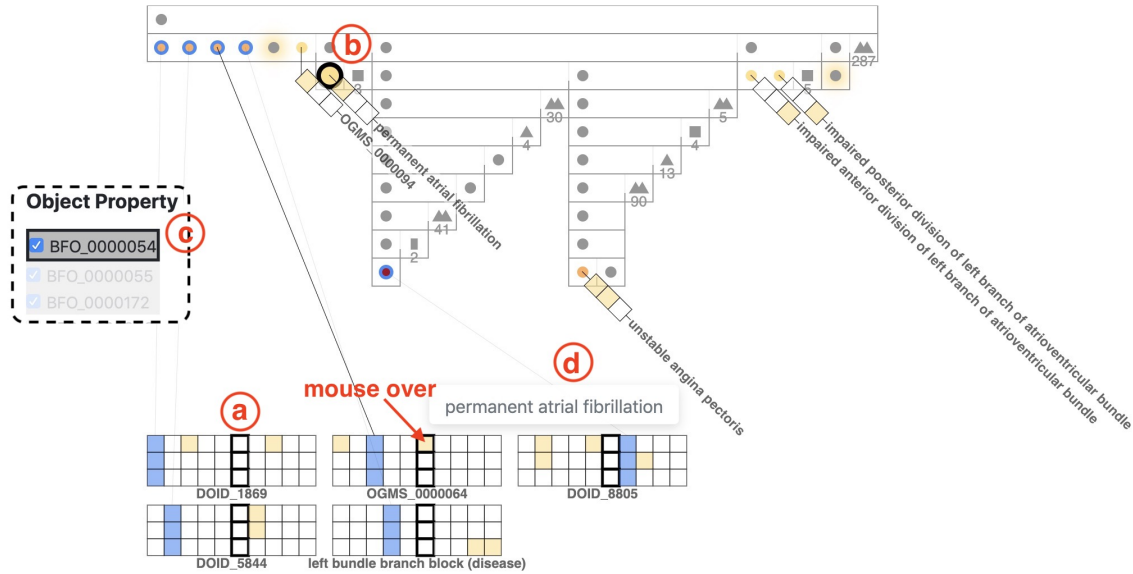


Figure 6.11: Highlighting is displayed when users hover the mouse over a cell on a grid label. Note, the leader line of that grid label is darkened in the visualisation. Others are faintly visible.

#### 6.1.2.4 Show Common Node

While the grid labels display all the association classes for each selected class, there is a need to easily identify the common association classes for all the selected classes (U14, see Table 2.5).

The system supports this by providing a “SHOW COMMON NODE” button (see Figure 6.12 (a)) on the notification bar (introduced in Section 5.3.2.2). When users select multiple classes and click this button, the visualisation will be updated and only the classes that have the selected properties with all the selected classes will be highlighted with colours and labels (shown in Figure 6.12). The grid labels are hidden in this mode.

#### 6.1.2.5 Multi-Focus Mode

As described in Section 5.3.2.2, the visualisation provides a *focus mode* to compress the hierarchy and show only the association classes for the selected class. In the new visualisation interface, the “FOCUS MODE” button is moved to the notification bar (see Figure 6.13 (a)) from its original location in the pop-up tooltip.

If multiple classes are selected, all the selected classes are used to recompute the *interesting* and *uninteresting* parts of the hierarchy (introduced in Section 4.2.2). The visualisation goes into a *Multi-focus mode* (see Figure 6.13). This gives emphasis on the associations for classes of interest and provides a compact view to bring these heterogeneous associations together (U12, U13, see Table 2.5).

#### 6.1.2.6 Search Function Refinement

As mentioned in Section 5.3.2.3, the visualisation supports search. When users enter a term in the search field, the matched class labels in the ontology will be displayed as a

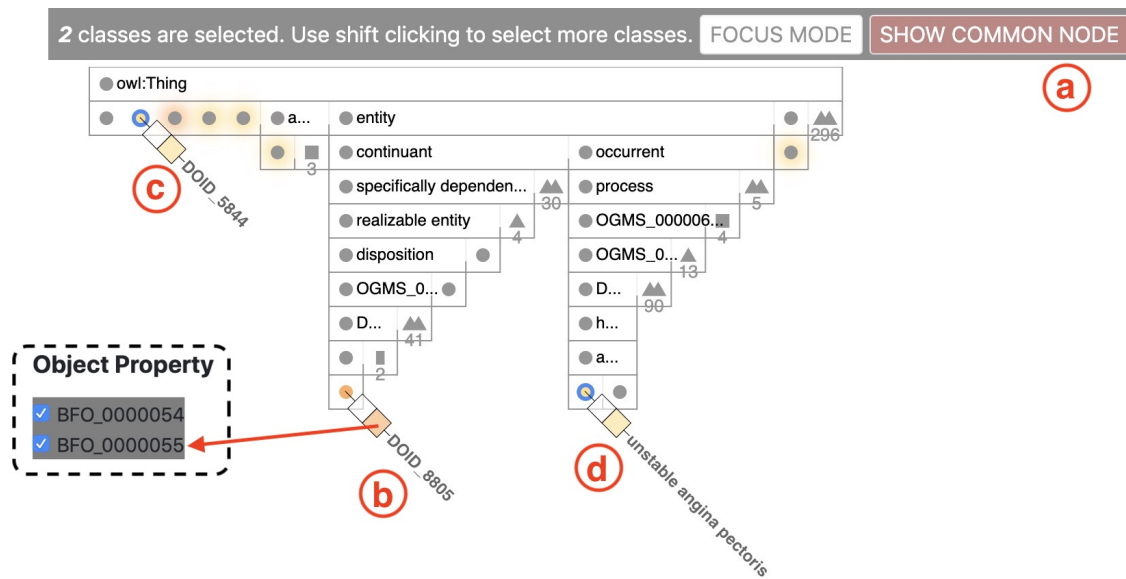


Figure 6.12: The *show common node* mode identifies the common node ⑥ for the selected classes ③ and ④, indicating ⑥ is linked by one of the selected properties *BFO\_0000055* (see arrow) to both ③ and ④.

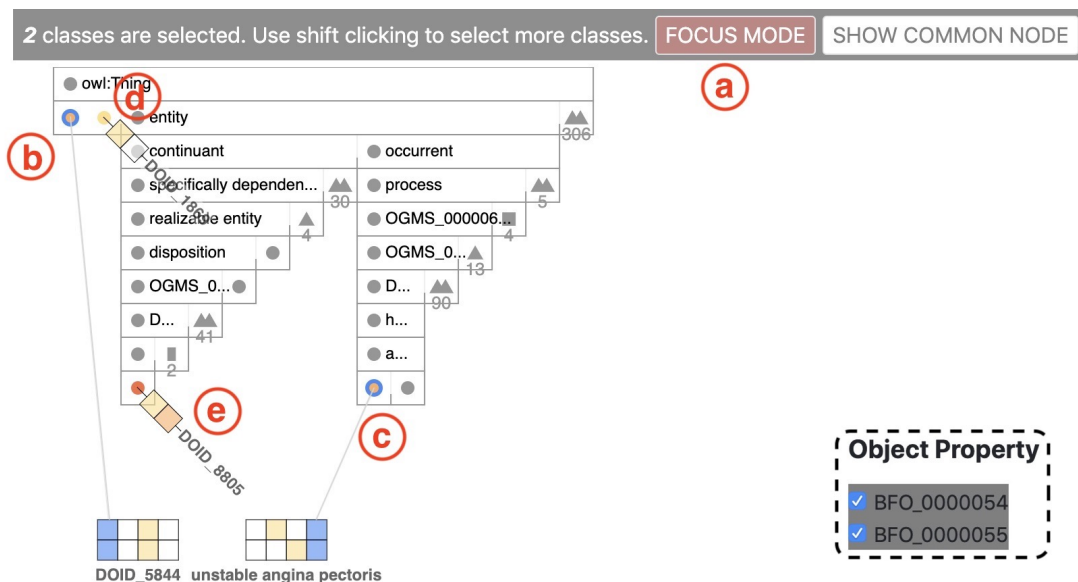


Figure 6.13: The *multi-focus mode* for the selected classes ③ and ④. In *focus mode* the visualisation is compressed and only shows the association classes ④ and ⑤ for ③ and ④ together with their ancestor classes.



scrollable list. If users click a class label, that class will be selected and the visualisation will be updated to show associations of that selected class. The view will also be centred on that class.

To support multiple selection of classes, this search function is refined to draw a bouncing arrow pointing to the searched class to indicate the location of that class, rather than making a selection of it as in the previous visualisation. This allows users to make a decision whether they want to select the searched class, while leaving their current selection as is.

## 6.2 Case Studies

This section demonstrates the capability of the new system to support experts to perform the analysis of associations in biomedical ontologies. Two case studies are presented to show how the new approaches facilitate the analysis involving multiple properties and classes to explore the associations. Case studies are widely used to evaluate the usability of new designs (Lam et al., 2011; Sedlmair et al., 2012; Wagner et al., 2018; Nobre et al., 2018). These walk-through case studies provide strong evidence for the satisfaction of the design requirements of OntoPlot and the usability of this interactive system to support the complex user tasks, showing the practical value of OntoPlot, though the need for training is acknowledged. The host-microbiome interactions ontology (He et al., 2019) and the drug adverse events ontologies (Guo et al., 2016; Wang et al., 2017) are used in the case studies that are chosen based on the discussions in the interviews, where analyses are similar to those in the papers.

### 6.2.1 Comparing properties to explore associations between microbiomes and diseases

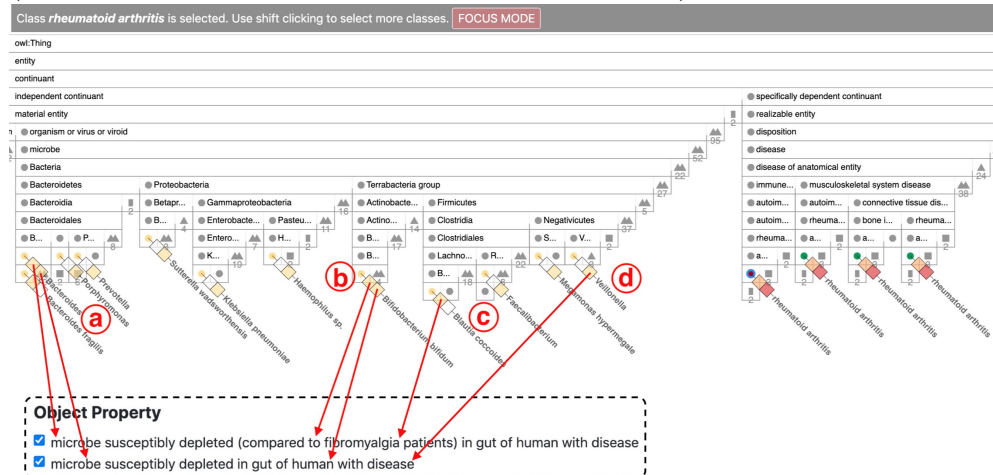
A microbiome is the aggregate of microbes living in an organism. It impacts its host's biological processes and is associated with diseases. The ontology of host-microbiome interactions (OHMI) (He et al., 2019) describes the associations between hosts, microbiomes and diseases. The analysis process of increase or decrease interactions of microbes in different diseased hosts is demonstrated here using OHMI in OntoPlot. OntoPlot supports experts to explore this OHMI ontology and to confirm or disprove their hypotheses.

#### 6.2.1.1 Comparing microbe interactions in different diseased patient guts

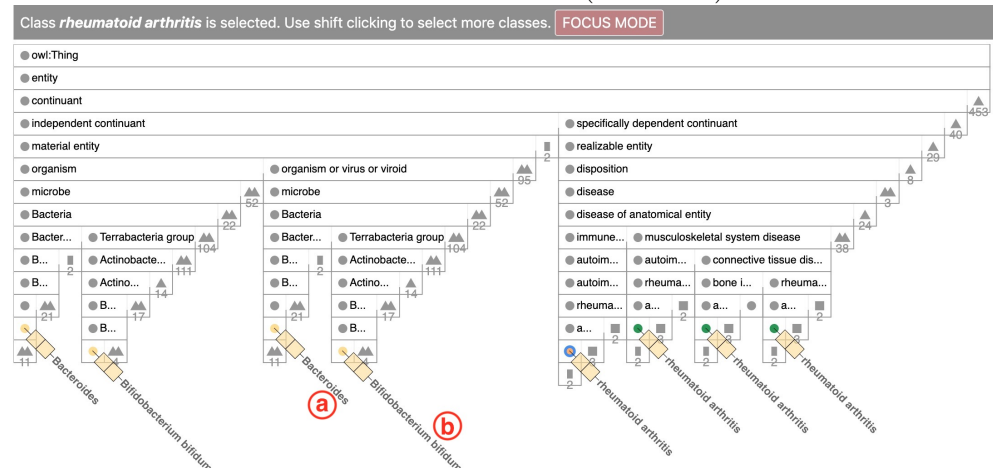
Firstly, after browsing the property list, the expert finds a property *microbe susceptibly depleted (compared to fibromyalgia patients) in gut of human with disease*, which compares the decrease of microbes in the guts of fibromyalgia patients with other disease patients, is more specific than another property *microbe susceptibly depleted in gut of human with disease*.



**Step 5.** In the *focus mode* (shown below), the expert can easily spot that there are several microbes linked either by the *microbe susceptibly depleted (compared to fibromyalgia patients) in gut of human with disease* property (for example, ㉓) or *microbe susceptibly depleted in gut of human with disease* property (for example, ㉑), and also some microbes linked by both properties, i.e., having the two label boxes coloured, indicating they are *intersection* classes (explained in Section 6.1.1.3, for example, ㉒ and ㉔).



**Step 6.** The expert then wants to see only the *intersection* classes, so selects the *intersection* operation. Then the visualisation is redrawn to produce the *focus mode* of *intersection* associations for the *rheumatoid arthritis* class (see below).



This visualisation clearly shows two bacteria *Bacteroides* (㉑) and *Bifidobacterium bifidum* (㉒) linked by both selected properties to the disease *rheumatoid arthritis*.

**Step 7.** The expert notes that *rheumatoid arthritis* patients had *Bacteroides* and *Bifidobacterium bifidum* bacteria depleted in their guts. This is also consistent when comparing to *fibromyalgia* patients.



Step 5.

The resulting visualisation is shown below.

Each grid label shows the association situation for each microbe class. In the grid labels, each column represents a disease class, and each row represents a selected property.

The expert quickly identifies from the grid labels that there is no microbe expanded in human guts and mouse or rat guts with the same disease, as there is no column with more than one cell coloured.

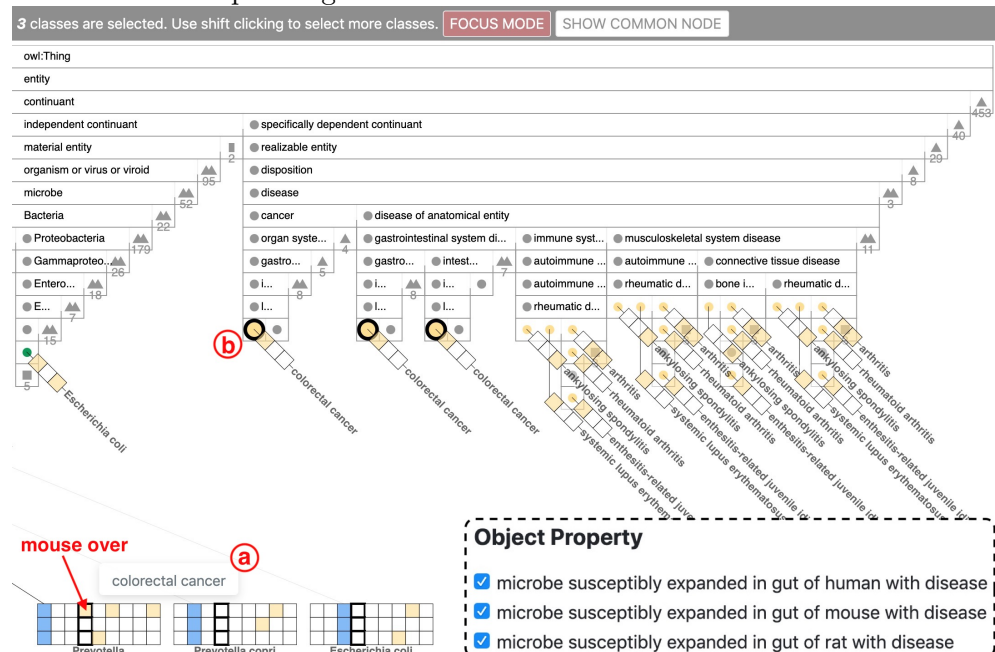
Step 6.

Then the expert clicks the “FOCUS MODE” button and goes into the *multi-focus mode* for the selected classes.

Step 7.

As the visualisation is still wider than a single screen, the expert cannot fully observe the whole disease subtree. The expert clicks the cell of colorectal cancer in the grid label, and the visualisation scrolls to the disease subtree.

**Step 8.** The whole disease subtree is shown below. Note, the three selected microbe classes are scrolled out of the view, but their grid labels are kept in the view, with leader lines pointing off screen.



The expert hovers the mouse cursor over each coloured cell in a grid label (for example, the colorectal cancer cell) and examines the information of each associated disease from the pop-up tooltip ① and the corresponding enlarged class circles ② in the hierarchy.

**Step 9.** After investigating each disease class in the grid labels, the expert notices that *Prevotella* expanded in colorectal cancer, systemic lupus erythematosus and enthesitis-related juvenile idiopathic arthritis human guts, and ankylosing spondylitis rat guts. *Prevotella copri* expanded in rheumatoid arthritis human guts and arthritis mouse guts. *Escherichia coli* expanded in rheumatoid arthritis human guts and arthritis rat guts.

**Step 10.** The expert notes that there is no common disease having microbes expanded in all three species, so would reject hypothesis 1. There are two microbes *Prevotella* and *Escherichia coli* expanded in human and rat guts but with different diseases, and one microbe *Prevotella copri* expanded in human and mouse guts but also with different diseases.

## Hypothesis 2

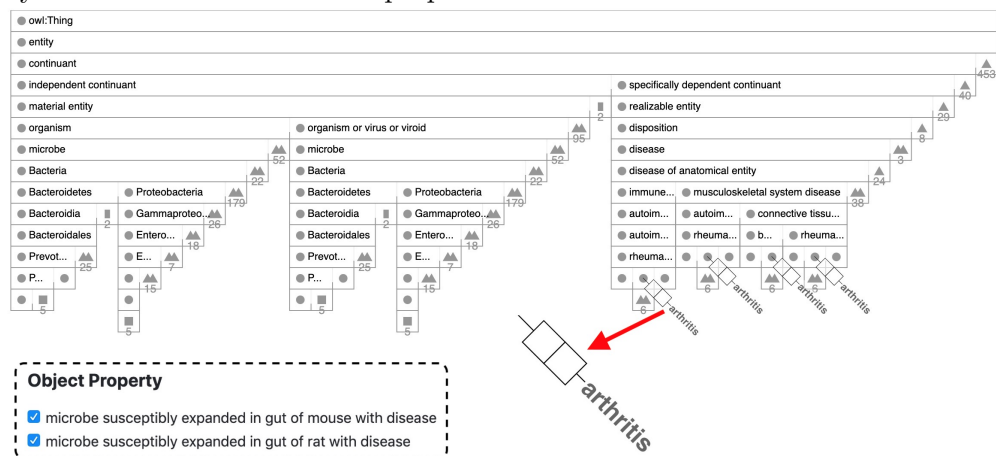
There might be the same host-microbiome interactions between mice and rats as they are similar species.

## Workflow 2

**Step 1.** The expert deselects the *microbe susceptibly expanded in gut of human with disease* property and the visualisation is recomputed.

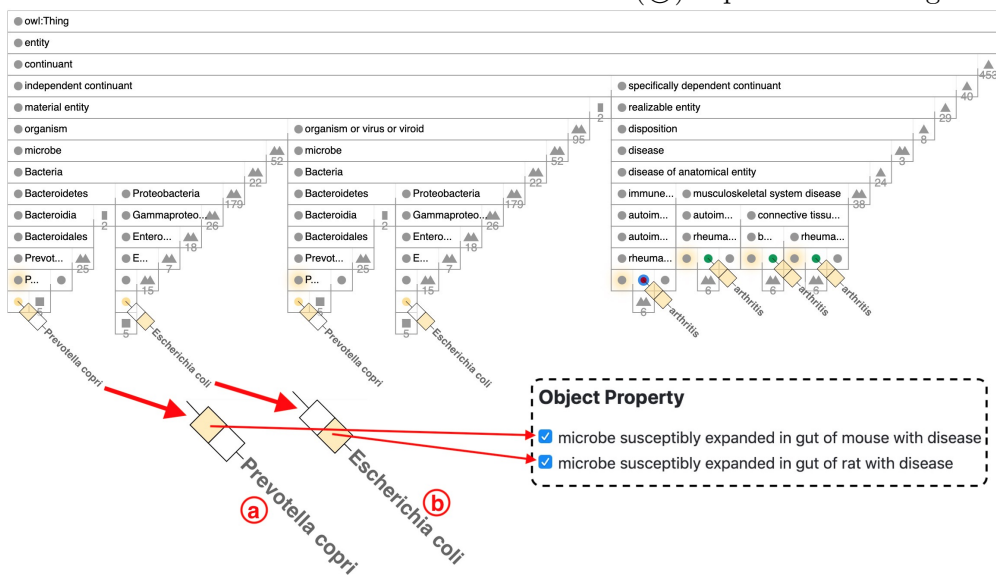
**Step 2.** This time the *intersection* operation is available, which indicates there is at least one common class of the *microbe susceptibly expanded in gut of mouse with disease* and *microbe susceptibly expanded in gut of rat with disease* properties, so the expert selects the *intersection* operation to see the common class(es).

**Step 3.** The expert notices from the resulting visualisation (shown below) that although there is a class **arthritis** disease that still remains labelled, none of the boxes on its label are coloured. This indicates that this *intersection* class **arthritis** disease has microbes expanded in both its diseased mouse and rat guts, but the microbes are different in different host species as none of the microbe classes is linked to it by both of these two selected properties.



**Step 4.** Then the expert selects the **arthritis** class and clicks the *union* operation, using the *union* mode to investigate its associated microbes.

**Step 5.** From the resulting visualisation shown below, the expert observes that the arthritic mice have *Prevotella copri* microbes (a) expanded in their guts, while arthritic rats have *Escherichia coli* microbes (b) expanded in their guts.





**Step 6.** The expert notes that there is a common disease **arthritis** having microbes expanded in its infected mouse or rat guts, but the microbes are different. There is no common microbe expanded in the guts of diseased mice and rats, so the expert would reject hypothesis 2. Hence, no common pair of host-microbiome interaction is found either among human, mouse and rat, or between any two of them.

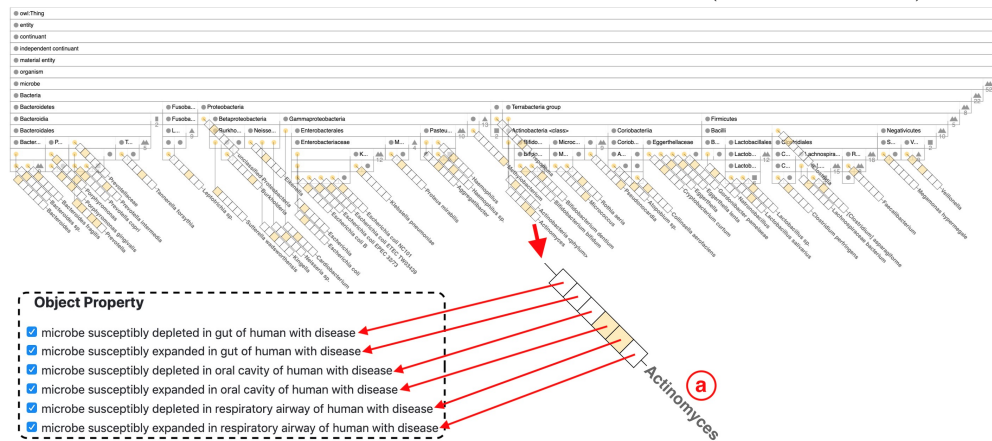
### 6.2.1.3 Comparing microbe interactions in different diseased human hosts

Finally, the expert wants to explore the ontology to compare multiple properties to see interactions in different human hosts.

#### Workflow

**Step 1.** The expert selects six properties, *microbe susceptibly depleted in gut of human with disease*, *microbe susceptibly expanded in gut of human with disease*, *microbe susceptibly depleted in oral cavity of human with disease*, *microbe susceptibly expanded in oral cavity of human with disease*, *microbe susceptibly depleted in respiratory airway of human with disease* and *microbe susceptibly expanded in respiratory airway of human with disease*, which might be comparable and interesting to explore together to see the host-microbiome interaction situations in three human organisms that this ontology mainly focuses on: human guts, oral cavities and respiratory airways.

**Step 2.** From the distribution of the coloured label boxes (shown below), the expert does not spot any microbes expanded or depleted in all these three hosts, as there is no microbe having all six label boxes coloured (for example, ①).





**Step 3.**

From the minimap (shown below), the expert notices that there are several red circles (Ⓐ), which represent the most associations, in the right part of the ontology that are not currently displayed on the main visualisation. Thus the expert hovers the mouse over these circles on the minimap and from the labels shown on the pop-up tooltips (for example, Ⓑ), the expert finds out these circles all represent the **rheumatoid arthritis** disease class and the duplication of the circles is caused by the multiple inheritance of this class.

mouse over

Ⓐ

Ⓑ

rheumatoid arthritis

SHOW ASSOCIATION LABEL

SHOW PATH LABEL

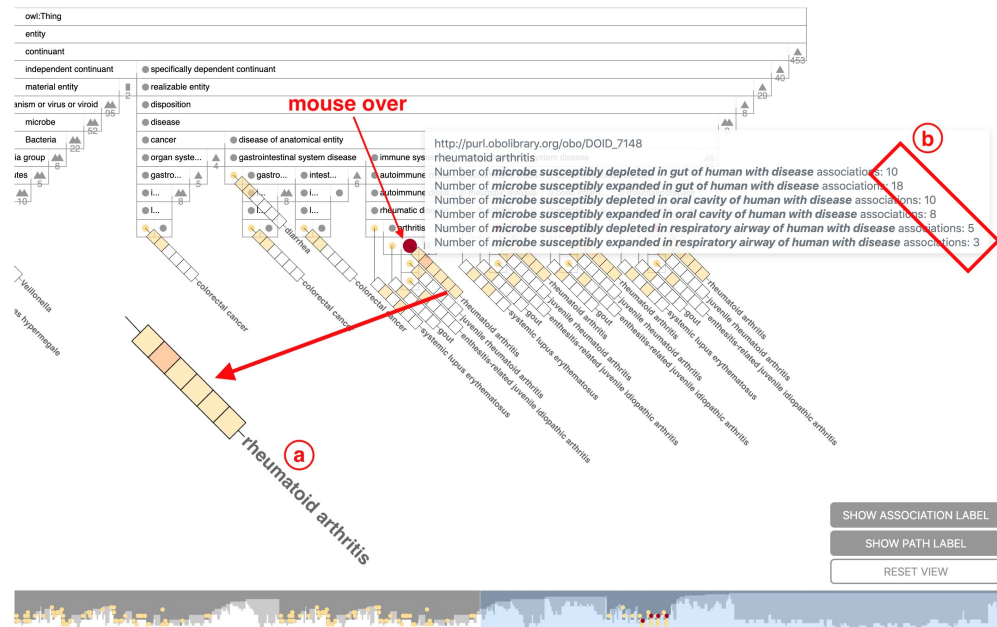
RESET VIEW

**Step 4.**

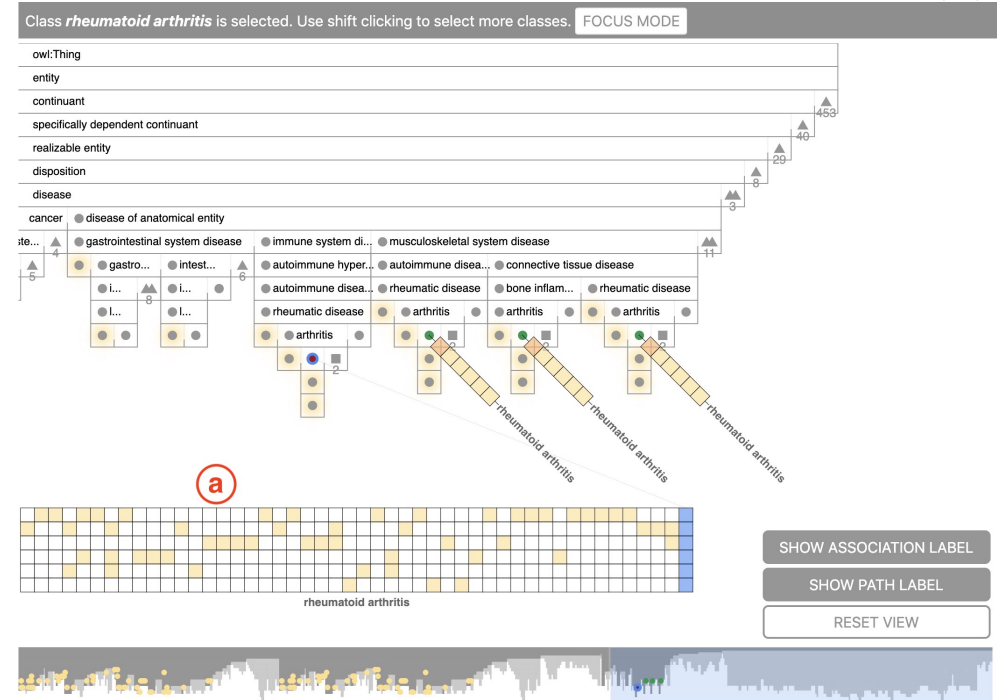
The expert clicks one of the circles on the minimap to scroll the main visualisation to the location of that class.

**Step 5.** Then the expert finds out the disease **rheumatoid arthritis** (a) has microbes interacting with all three hosts when the hosts are infected by this disease, as all six label boxes of it are coloured.

The expert moves the mouse over this disease and notes the number of associations of this disease for each selected property from the pop-up tooltip (b).



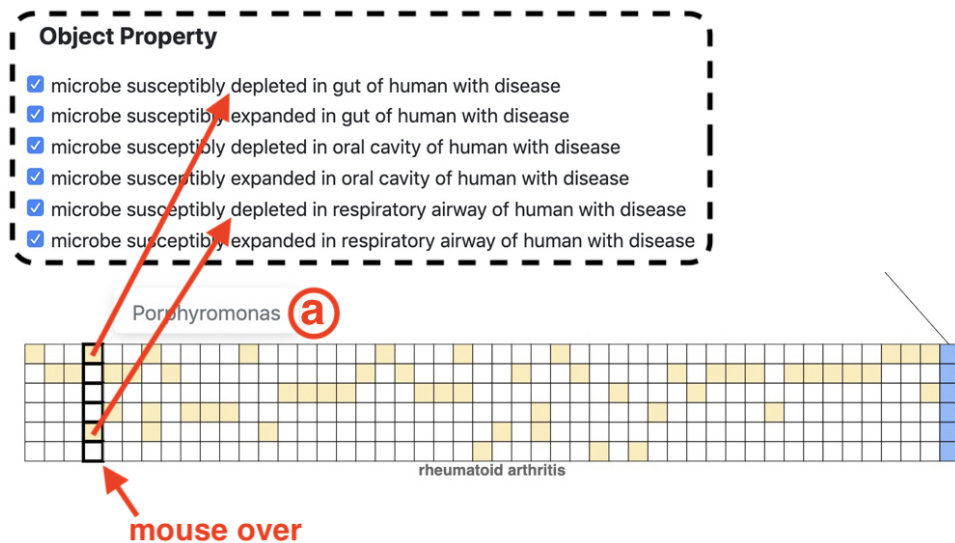
**Step 6.** In order to see the distribution of these associations better, the expert shift-clicks the rheumatoid arthritis class and a grid label is generated for it (a).



**Step 7.** From the grid label (shown below), the expert directly observes the situation of expansion or depletion of each microbe in rheumatoid arthritis patients' organs.

The expert notes that there are some microbes with more than one cell coloured, indicating there is more than one interaction with the diseased hosts, so the expert hovers on the columns to get the labels of the microbes from the pop-up tooltip.

For example, *Porphyromonas* (a) is depleted in rheumatoid arthritis patients' guts and respiratory airways.



**Step 8.** The expert deselects the rheumatoid arthritis class to reset the visualisation and investigate other classes.

- Step 9.

After browsing the microbe subtree, the expert observes that some sibling microbe classes have the same properties applied to them.

In order to figure out what the associated diseases are for each microbe and whether there are any diseases common to the siblings, the expert uses shift-clicking to select the sibling microbes and clicks the “SHOW COMMON NODE” button. Then the expert clicks the “FOCUS MODE” button to compress *uninteresting* subtrees.

The figure below shows an example of the resulting visualisations, showing common nodes for the sibling microbes Eikenella, Kingella and Neisseria sp. (a) in the *multi-focus mode*.

The screenshot shows a complex visualization interface. At the top, a grey bar contains the text "3 classes are selected. Use shift clicking to select more classes." and two buttons: "FOCUS MODE" and "SHOW COMMON NODE". Below this is a hierarchical tree structure. The left column lists entity types: entity, continuant, independent continuant, material entity, organism or virus or viroid, microbe, Bacteria, Proteobacteria, Betaproteobacteria, and Neisseria. The right column lists specific properties: specifically dependent continuant, realizable entity, disposition, disease, disease of anatomical entity, immune..., musculoskeletal system disease, autoimmune..., autoimmune..., and connective tissue dis... Each node in the tree has a small triangle icon and a number. A red arrow labeled "mouse over" points to a colored box in the hierarchy. A red circle labeled "a" highlights a specific node. A red circle labeled "b" highlights a tooltip box that appears when the mouse hovers over a colored box. The tooltip box contains the text "Number of microbe susceptible depleted in oral cavity of human with disease associations: 1".

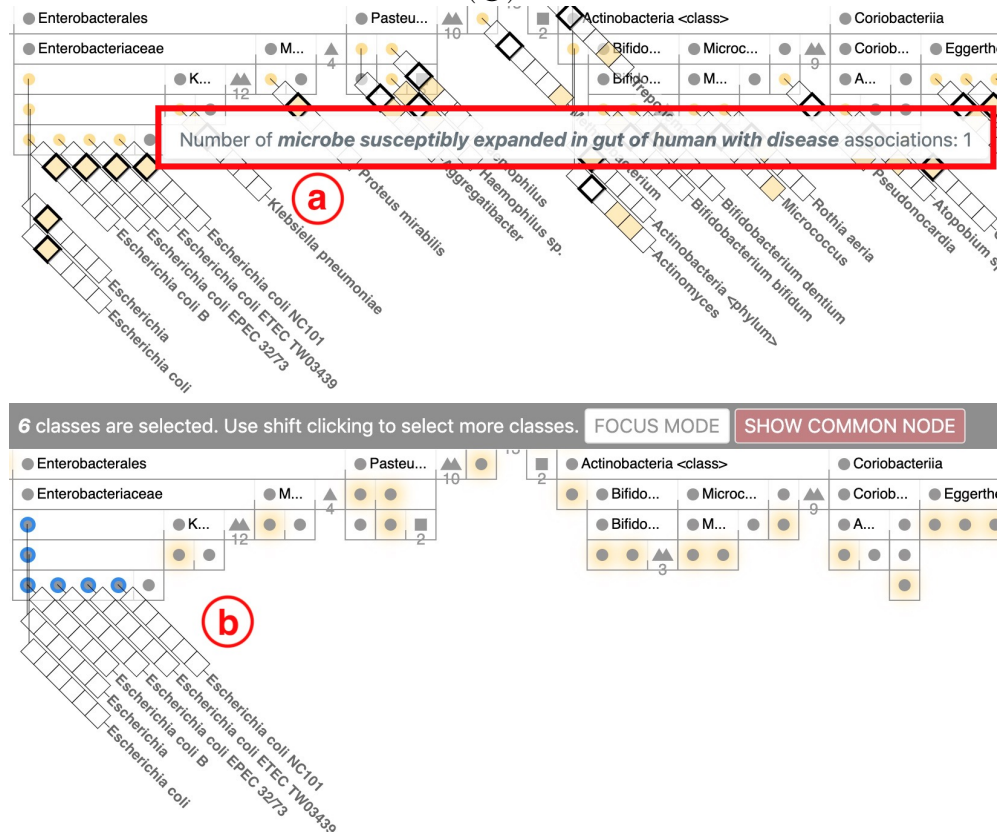
The expert hovers the mouse over the coloured label box to get the information of the property that the box represents from the pop-up tooltip (b). Consequently, all the boxes on the association labels of different classes for the same property are highlighted.

The expert notes that *rheumatoid arthritis* diseased patients have the sibling microbes *Eikenella*, *Kingella* and *Neisseria* sp. depleted in their oral cavities.

Step 10.

Similarly, the expert notes that the sibling classes *Bacteroides* sp. and *Bacteroides fragilis* have one common disease *rheumatoid arthritis* for the *microbe susceptible expanded in gut of human with disease* property, indicating both microbes expanded in the *rheumatoid arthritis* patients’ guts.

**Step 11.** The expert also notices that the *Escherichia* group microbes (the six ancestor-descendant classes) all have the same property *microbe susceptibly expanded in gut of human with disease* applied to them (Ⓐ), but there is no common disease found after clicking the “SHOW COMMON NODE” button as all the label boxes turn to white (Ⓑ).



**Step 12.** The expert notes that the disease rheumatoid arthritis has microbes interacting in all the three diseased hosts: human guts, oral cavities and respiratory airways, but the microbes are different. No microbe has been found depleted or expanded in all the three hosts with any diseases. The sibling microbes *Eikenella*, *Kingella* and *Neisseria* sp., and another sibling microbes *Bacteroides* sp. and *Bacteroides fragilis* had the same host-microbiome interaction with a disease.

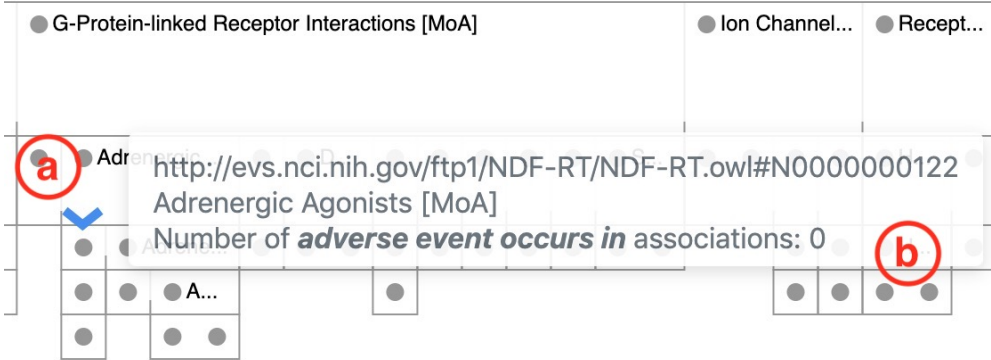
### 6.2.2 Linking properties to explore associations between drugs and their adverse events

An administration of a drug is a medical intervention to relieve illness. Following a medical intervention, unintended consequences like adverse events may occur. Two ontologies, the Ontology of Drug Neuropathy Adverse Events (ODNAE) that describes the neuropathy adverse events and their associated drugs (Guo et al., 2016), and the Ontology of Cardiovascular Drug Adverse Events (OCVDAE) that describes the adverse events of cardiovascular drugs (Wang et al., 2017), are presented here to demonstrate the analysis processes of drugs and their adverse events using OntoPlot.

6.2.2.1    **Analysing agonist and antagonist neuropathy-inducing drugs and their targets in ODNAE**

Certain drugs that act as agonists or antagonists of neurotransmitters contribute to the neuropathy adverse events. To investigate how the drugs induce neuropathy, the expert starts by analysing the agonist and antagonist drugs and their targets in the ODNAE knowledge base.

**Workflow**

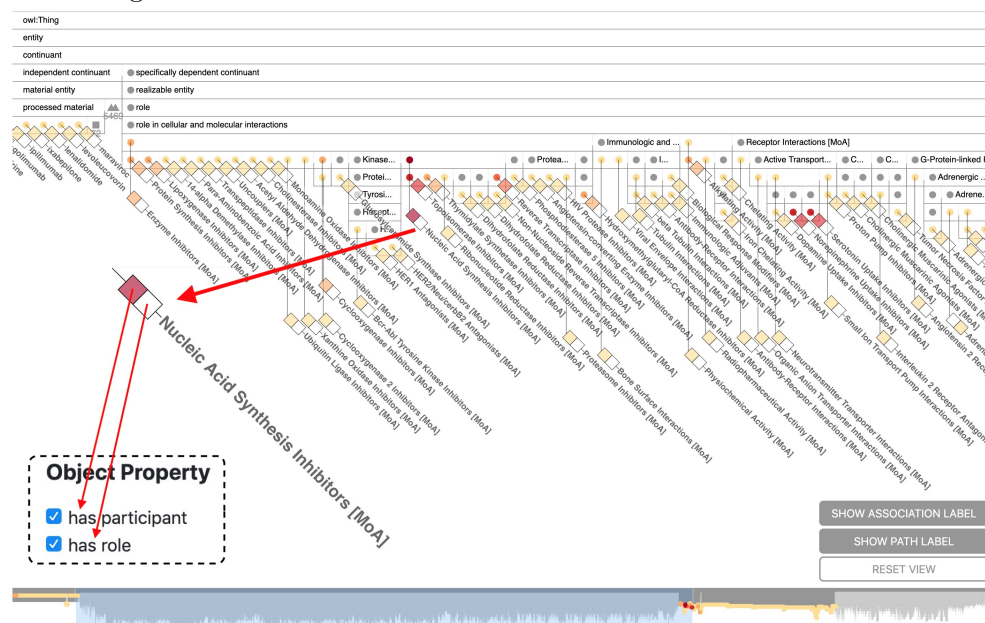
<b>Step 1.</b>	The expert selects the ODNAE ontology. By default, the first property <i>adverse event occurs in</i> is selected to draw the visualisation.
<b>Step 2.</b>	After the visualisation is loaded, the expert searches the term “agonists” and gets a list of classes whose name contains the term “agonists”.
<b>Step 3.</b>	The expert browses the results list and finds some interesting classes that are related to the neurotransmitters such as “adrenergic”, “dopamine”, “hormone” and “serotonin”.
<b>Step 4.</b>	<p>The expert selects these interesting classes one-by-one to investigate their associations.</p> <p>For example, the expert clicks “Adrenergic Agonists” in the search result list. As shown below, the visualisation jumps to the Adrenergic Agonists class and a bouncing arrow is drawn pointing to that class ① with a pop-up tooltip ② showing its information.</p> <div></div>
<b>Step 5.</b>	<p>Then the expert clicks the Adrenergic Agonists class and its descendants such as Adrenergic alpha-Agonists [MoA] and Adrenergic alpha2-Agonists [MoA] classes to figure out what the associations are that these classes are involved in.</p> <p>OntoPlot helps this by moving the properties applied to the clicked class to the top of the property list and colouring the background of the properties grey.</p> <p>The expert also clicks the sibling class Adrenergic Antagonists [MoA] of Adrenergic Agonists and the descendants of Adrenergic Antagonists [MoA] such as Adrenergic alpha-Antagonists [MoA] and Adrenergic beta-Antagonists [MoA] for the same purpose.</p>

**Step 6.** Same exploration happens for the Dopamine Agonists, Hormone Receptor Agonists and Serotonin Agonists classes that are on the search results list.

**Step 7.** From the exploration mentioned above, the expert finds out that two properties *has participant* and *has role* are used interchangeably to link the agonist or antagonist drugs to their targets.

**Step 8.** Then the expert selects these two properties and deselects the *adverse event occurs in* property to redraw the visualisation.

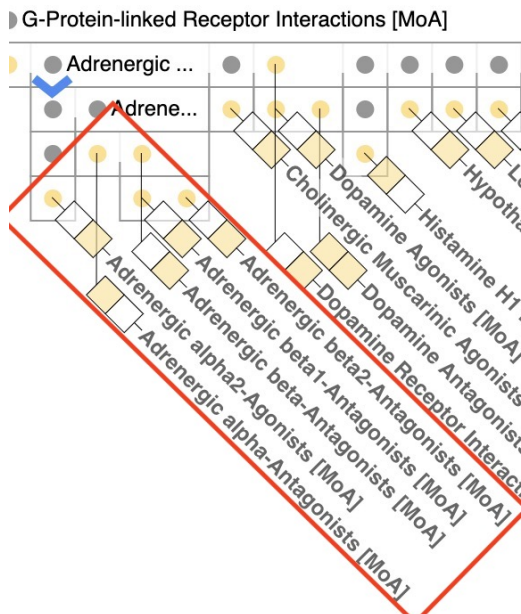
**Step 9.** The new generated visualisation is shown below.



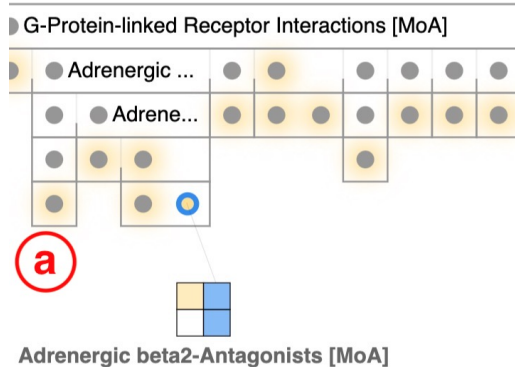
From the visualisation, the expert observes there are lots of classes involved in the agonist and antagonist interactions.



- Step 10.** To locate the interesting adrenergic group classes, the expert clicks “Adrenergic Agonists” in the search results list again.
- From the visualisation shown below, the expert observes there is one adrenergic agonist and four adrenergic antagonists present in the ontology.



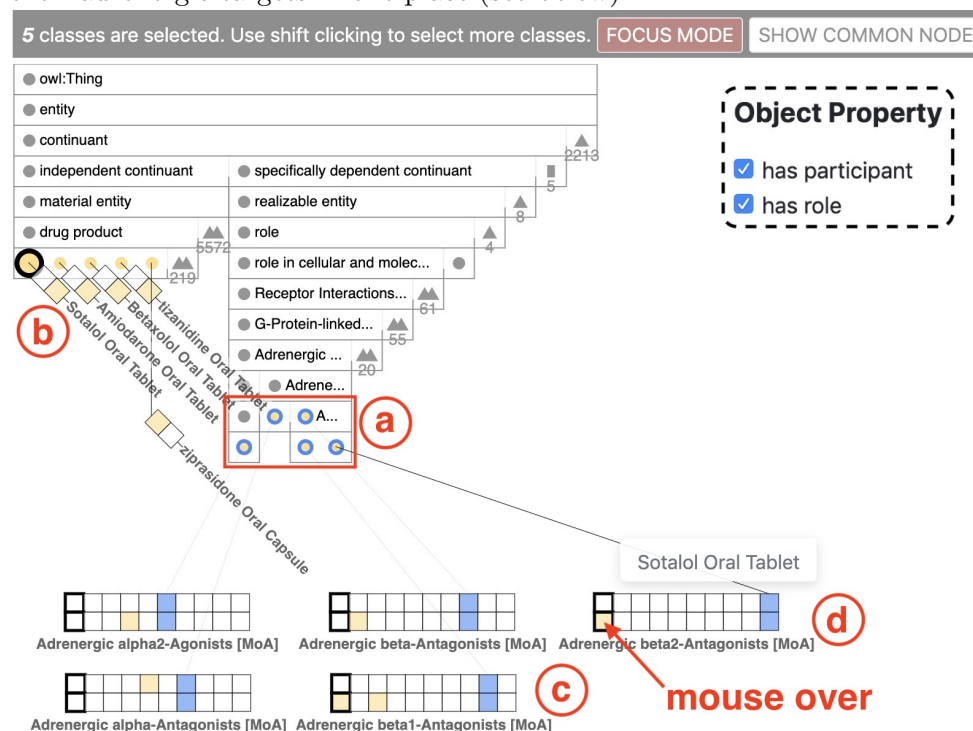
- Step 11.** In order to investigate what the drugs are for each adrenergic agonist or antagonist, the expert shift-clicks these classes one-by-one to get their grid labels.
- OntoPlot guides this process by drawing a colour glow surrounding the classes that have associations (for example, @) after selecting any classes, to give a hint indicating where the other association classes are.





**Step 12.** After selecting all the adrenergic group classes, the expert clicks the “FOCUS MODE” button.

Then the visualisation goes into the *multi-focus mode* of the selected adrenergic group classes (Ⓐ), showing the agonist and antagonist drugs and their adrenergic targets in one place (see below).



The expert hovers over each cell in the grid labels to investigate how the drugs associate with each target.

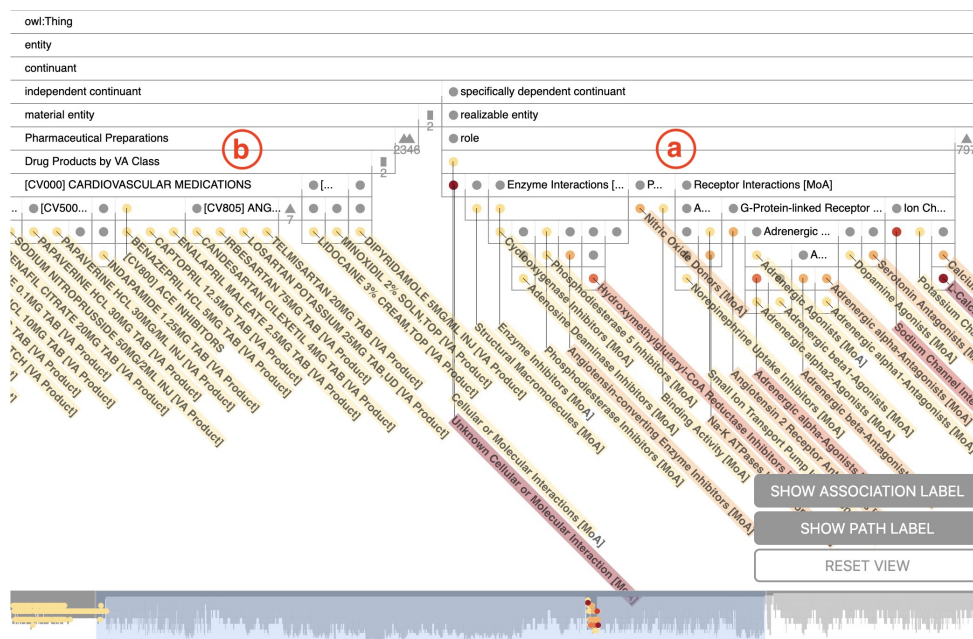
For example, the highlighting of Sotalol Oral Tablet class (Ⓑ) and the coloured cell in the grid labels of adrenergic beta1 (Ⓒ) and adrenergic beta2 (Ⓓ), shows the drug Sotalol Oral Tablet is the antagonist of both adrenergic beta1 and adrenergic beta2.

**Step 13.** The expert repeats the same process to find out the inducing agonist and antagonist drugs for the other interesting neurotransmitters: dopamine, serotonin and hormone.

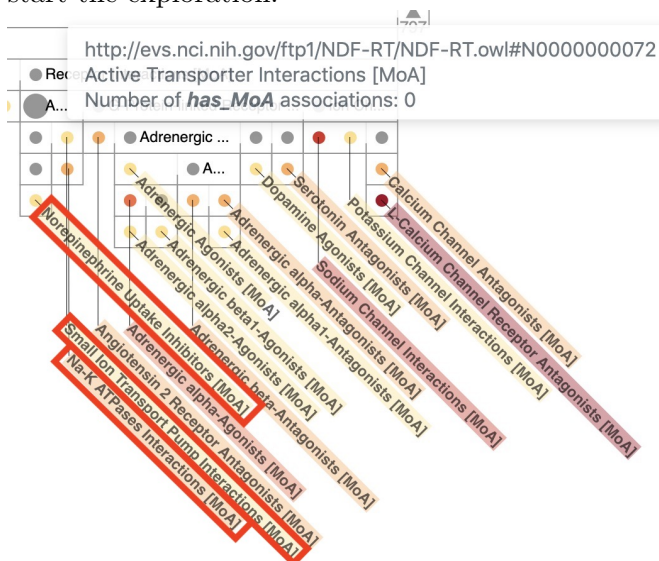
### 6.2.2.2 Analysing cardiovascular drugs and their adverse events based on drug mechanism of action classification in OCVD AE

A mechanism of action (MoA) is a biochemical interaction through which a drug produces its effects. Classifying drugs under MoA is an effective way to bind drugs to the interactions they trigger. In OCVD AE, the expert wants to find out whether any cardiovascular drugs that are under a common MoA cause the same adverse events.

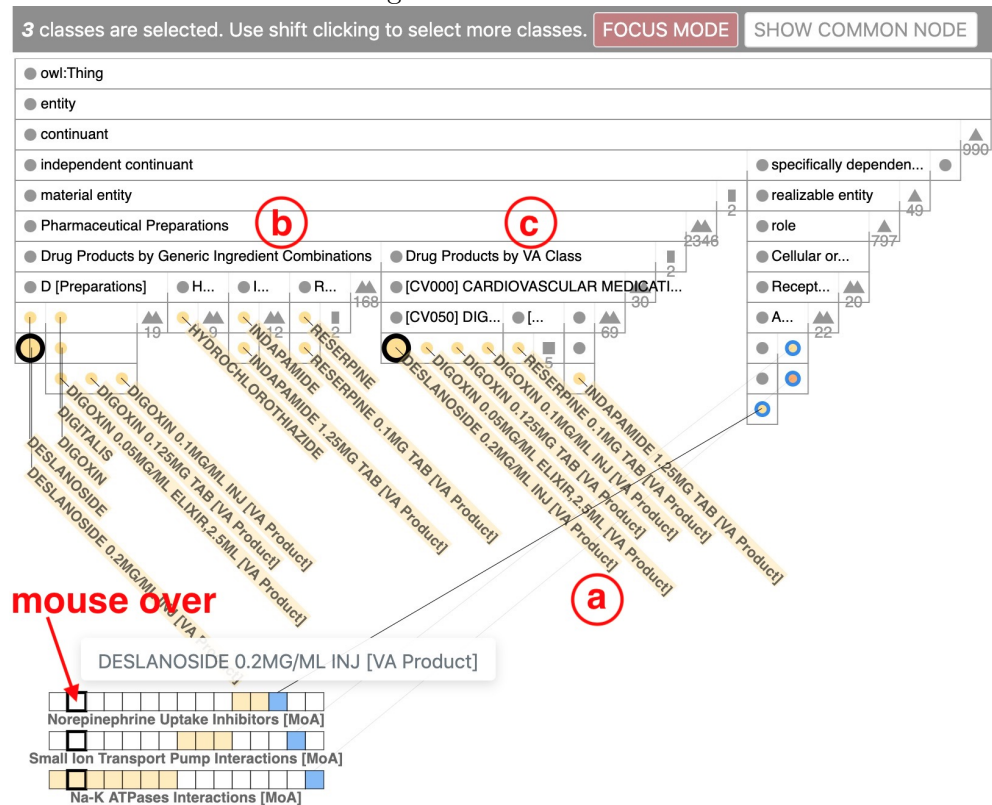
<b>Step 1.</b>	<p>The expert selects the OCVDAE ontology.</p> <p>Then the expert finds the property <i>has_MoA</i> might be used for the MoA classes, so the expert selects this property to generate the visualisation.</p>
<b>Step 2.</b>	<p>After browsing the visualisation as shown below, the expert confirms <i>has_MoA</i> is used to link the MoA classes (Ⓐ) and the drug products (Ⓑ), which is what the expert wants.</p>



**Step 3.** Then the expert chooses three interactions Norepinephrine Uptake Inhibitors [MoA], Small Ion Transport Pump Interactions [MoA] and Na-K ATPases Interactions [MoA] from the Active Transporter Interactions [MoA] group to start the exploration.



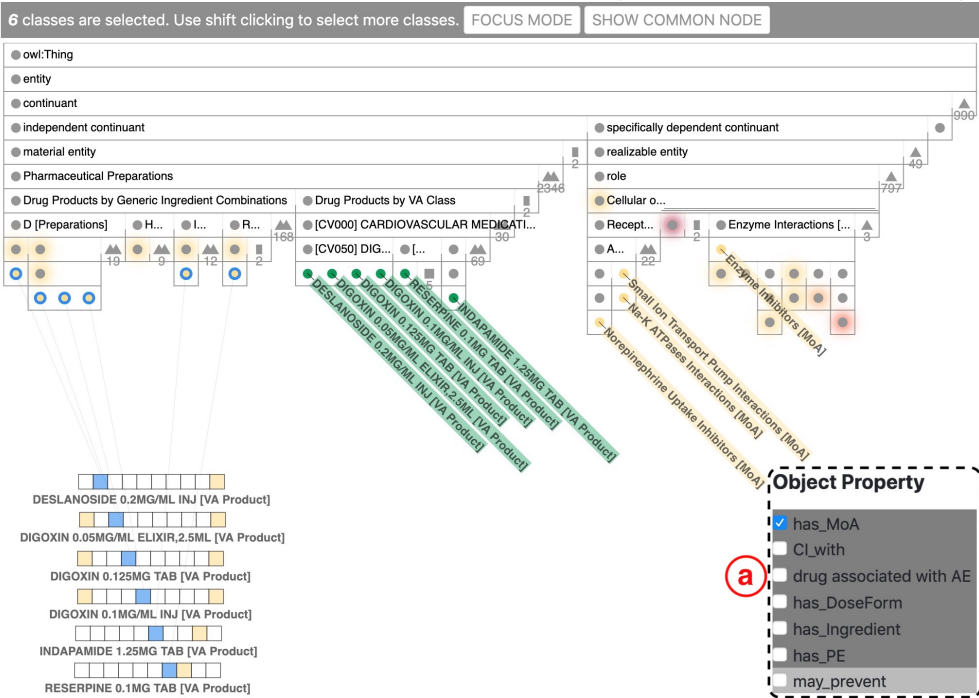
**Step 4.** The expert shift-clicks these three interactions and clicks the “FOCUS MODE” button. The resulting visualisation is shown below.



The expert then hovers the mouse over the cells in the grid labels to find out the drugs associated with each interaction.

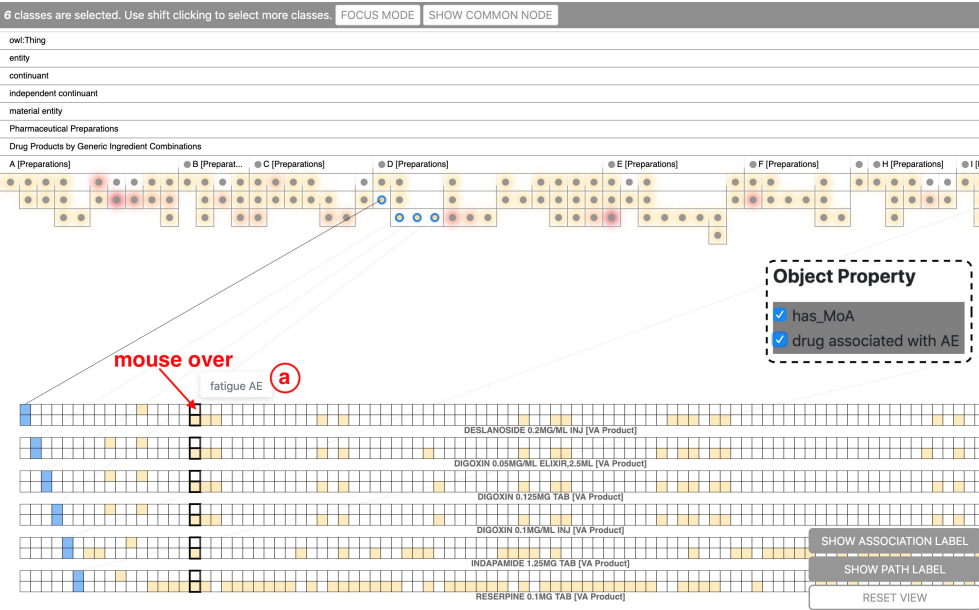
The expert observes that the leaf classes (for example, (a)), which are the final drug substances, are duplicated and grouped under two different categories, Drug Products by Generic Ingredient Combinations (b) and Drug Products by VA Class (c).

**Step 5.** The expert shift-clicks these leaf drug classes to select them. The resulting visualisation is shown below. Note, the green association labels indicate these classes are identical classes to the selected classes (due to multiple inheritance).



From the updated property list, the expert finds out one property *drug associated with AE* (a) is commonly applied to these selected drugs and might be used to link the drugs and the adverse events.

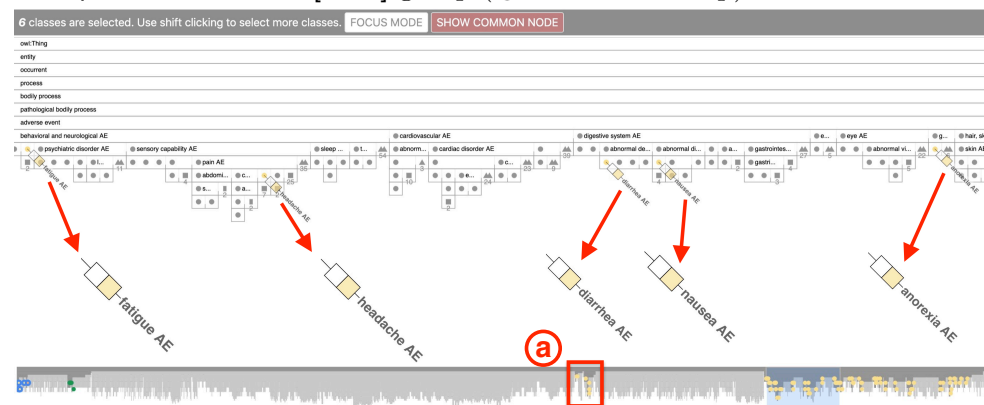
**Step 6.** The expert selects the *drug associated with AE* property to add it to the visualisation. The resulting visualisation is shown below.



Then the expert hovers over the grid cells and confirms this property links the drugs and the adverse events (for example, fatigue AE a).

**Step 7.** To see only the common adverse events for these six drugs, the expert clicks the “SHOW COMMON NODE” button.

From the resulting visualisation (shown below), the expert notes five adverse events fatigue AE, headache AE, diarrhea AE, nausea AE and anorexia AE are associated with all the selected drugs, which are classified under the Active Transporter Interactions [MoA] group (@ on the minimap).



## 6.3 Summary

Figure 6.14 summarises the key features of OntoPlot used in the two case studies. This figure organises these features that are designed for heterogeneous association tasks based on the workflow of each use case. Other general functions like search, minimap or pop-up tooltips are not included in this outline.

As each row in this summary figure is only for one feature, the similarities and differences between use cases can be discovered. After selecting interesting properties, experts are usually interested in particular classes. They make the selection of classes, then investigate the distribution of the associations for the selected classes using grid labels and the *focus mode*. They also use the “show common node” function to get the commonly associated classes for the selected classes. The property *intersection* operation is used either to get intersection classes for interesting properties or to get the intersection classes for interesting properties and classes. Experts can also explore more properties or classes at any stage of the analysis.

## 6.4 Discussion

As described in the case studies, experts use multiple properties in various ways to perform analysis. Depending on the different ontology structures, experts need to compare properties (Section 6.2.1), find complementary information from different properties (Section 6.2.2.1) or use multiple properties as a chain to link different groups of classes (Section 6.2.2.2).

The OntoPlot revisions that are designed for visualising multiple properties support these diverse user needs. The visualisation provides association labels and grid labels, with

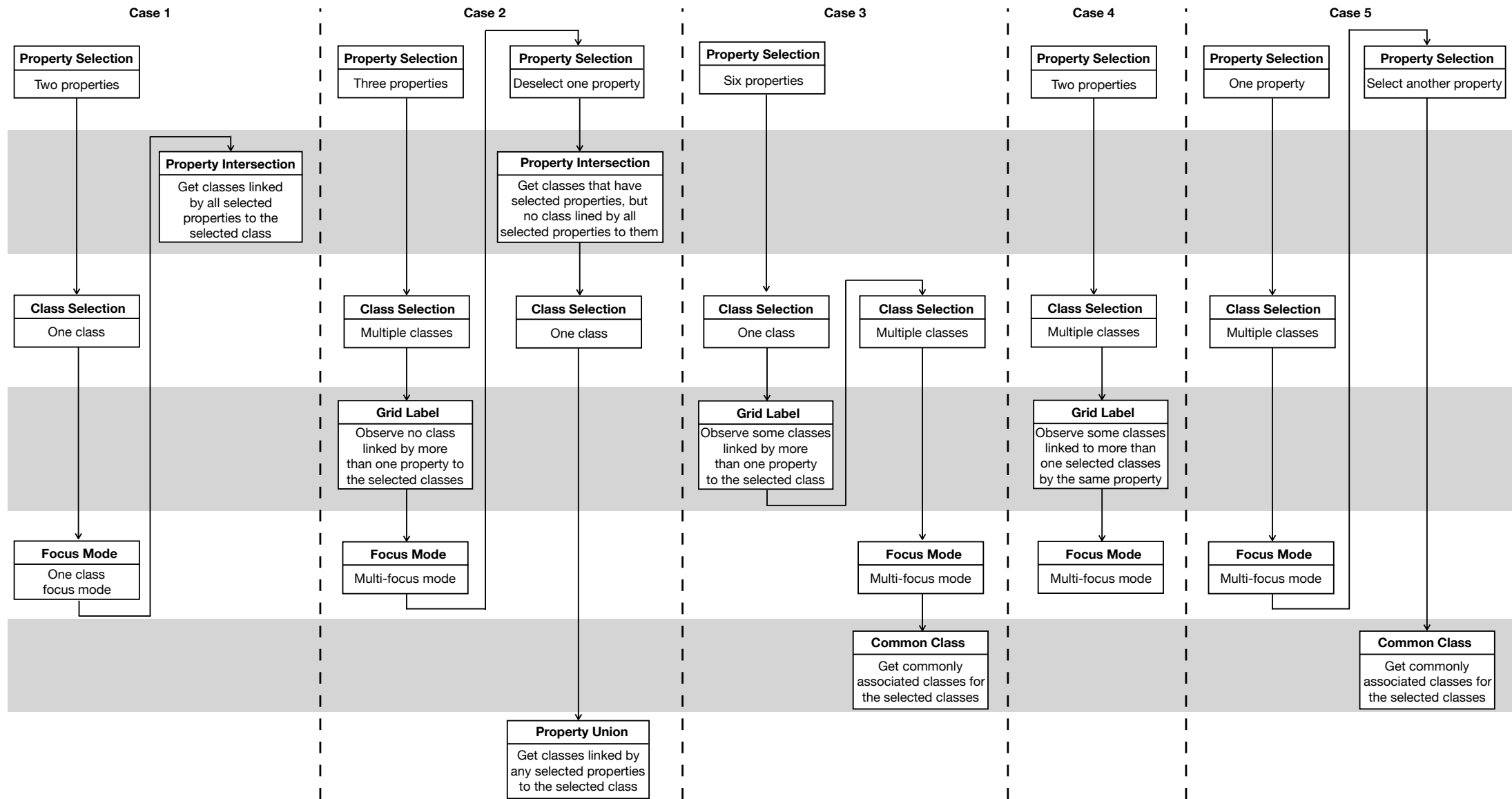


Figure 6.14: Summary of key features of OntoPlot for heterogeneous association use cases.

each box (or cell) representing an association that consists of a labelled class, an associated class and a property. These representations put the desired information together in a compact form, facilitating users to get an overview of the data and make comparisons. The interactions also ease the analysis process by finding and filtering information for users, such as double-clicking a grid label cell to scroll the view and locate a class, and double-clicking an association label box to hide unrelated associations.

As discussed in Chapter 2, currently these types of analyses sometimes need to be done by querying the ontology using SPARQL queries. For example, for the case described in Section 6.2.2.1, Guo et al. (2016) created SPARQL queries to identify agonist or antagonist drugs for each neurotransmitter. Similarly, He et al. (2019) wrote a number of SPARQL queries to retrieve the associated microbes for six different host profiles, which is the case demonstrated in Section 6.2.1.3. However, experts have stated that writing SPARQL queries requires significant effort, especially when complicated queries are needed. Experts need to understand the syntax and logic of the queries. Also, this kind of text-based query is error prone, requiring manual checking of results, which is time consuming. OntoPlot directly supports such analyses through interaction without the need for writing SPARQL queries.

For some other cases like the ones demonstrated in Section 6.2.1.3 and 6.2.2.1, experts are known to have manually checked each association in Protégé and visualised the associations by labelling them on the screenshot of the ontology hierarchy (He et al., 2019) as shown in Figure 2.10b or using tables (Wang et al., 2017). The cases that involve *union* and *intersection* of properties (Section 6.2.1.1, 6.2.1.2) or showing common node for classes (Section 6.2.1.3, 6.2.2.2) also are quite time consuming and error prone with manually checking. Manually checking is often impractical, especially for the cases that are supported by switching between *union* and *intersection* to check the associations for particular classes. OntoPlot makes these tasks much easier.

Given that sometimes the ontologies being used are previously unknown to experts, OntoPlot guides experts through the exploration process to find useful information, such as selecting related properties or classes. Highlighting applied to the property list to indicate the involved properties for selected classes and colour glow highlights around unselected classes are valuable approaches, as demonstrated in the use cases.

Although the case studies demonstrated in this chapter are specific to the bioinformatics domains, similar tasks need to be performed for most ontologies. Thus OntoPlot is applicable to ontologies from other domains, such as agronomy, e-government and bibliometrics, as discussed in Chapter 1. The visualisation also can be adapted to other hierarchical datasets to support equivalent cases, especially those involving non-hierarchical relations that need to be analysed.

## 6.5 Conclusion

This chapter discussed the approaches for visualising heterogeneous associations that involve multiple properties and classes in an ontology hierarchy. The visualisation encodes

this information in the association labels and grid labels, interactively supporting various user tasks and allowing analyses to be conducted more efficiently than with existing approaches to ontology visualisation.

The next chapter will summarise the work presented in this thesis and discuss the future research directions.



## Chapter 7

# Conclusions

Expressive ontologies contain rich information captured by class hierarchies and associations. Their large and complex structure demands effective visualisations to support user activities with such data. Most existing ontology visualisation systems emphasise ontology hierarchies, making it hard to find information about the non-hierarchical associations. Using these systems to perform analysis on ontology associations is tedious and time consuming. This is especially true for complex activities that involve diverse associations, which require considerable manual effort.

The research presented in this thesis was motivated by the needs for effective visualisations to support the exploration of large ontologies and associations.

### 7.1 Contributions

Chapter 4 addressed the first research question of how large ontologies can be effectively visualised. It presented a new visual representation, OntoPlot, for showing the hierarchy structure, achieving higher space-efficiency than basic hierarchy representations like node-link layered trees, indented lists and icicle plots, and depicting the hierarchical structure more clearly than treemaps. The chapter also introduced a set of glyphs allowing visual compression for data elements that are not of current interest to users, with different glyph shapes representing different compressed structures. It presented two user studies showing that OntoPlot can effectively visualise large ontology hierarchies to help the cognition and comprehension of their hierarchical structure.

To answer the second research question of how homogeneous associations can be effectively visualised in an ontology, Chapter 5 described approaches that clearly represent homogeneous ontology associations on top of the hierarchy and emphasise the significant associations. The concept of interest was used to automatically compute the *interesting* and *uninteresting* parts of the ontology based on user interests of properties and classes, and dynamically update these sections of the visualisation during user interactions. The user evaluations confirmed that OntoPlot clearly shows the homogeneous associations and satisfactorily supports the exploration of them in the ontology hierarchy, which is strongly supported by the statistically significant positive results from an expert user study.

Chapter 6 presented a representation for visualising complex heterogeneous associations consisting of diverse properties and classes, alongside the ontology hierarchy, to address the third research question of how heterogeneous associations can be effectively visualised in an ontology. These designs were implemented and demonstrated via OntoPlot, with well-defined interactivity to facilitate users to perform analyses on ontology hierarchies and associations. The described case studies demonstrated OntoPlot’s strong capability and usability for visualising heterogeneous associations and facilitating the complex user tasks.

## 7.2 Limitations

Although comprehensive research has been conducted to address the research problems, there are still some limitations to the work.

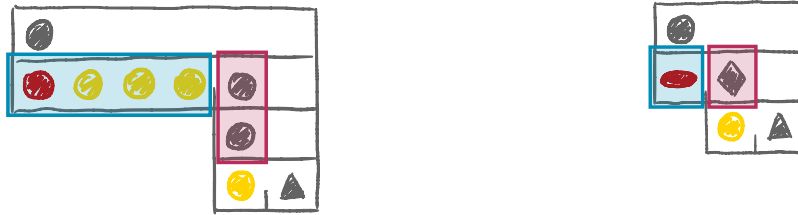
### 7.2.1 Evaluation

First, the designed representations were evaluated through user studies and case studies, which can be considered preliminary if put into the context of real working environments. Even though this research followed the user-centred design methodology and is user-driven, to fully assess the usability of the system, it would be worthwhile to do more testing and observations of expert users utilising OntoPlot for their analysis tasks. Further research in this direction is planned.

### 7.2.2 Scalability

Secondly, the scalability of the proposed visualisation is still limited. Very large ontologies that have tens or hundreds of thousands of classes and associations impose computational challenges to the system and scalability issues to the visualisation. To facilitate easy overview and browsing of such large data, further compression might be a solution. For example, another two possible cases that could be compressed are sibling leaf association classes and intermediate level classes that do not have any associations. As illustrated in Figure 7.1, two glyphs, such as ellipse and diamond, could be introduced to visually summarise these two cases. This would save more space that could be used to accommodate additional data elements on screen. Interactions can be employed to allow exploration of these compressed parts of hierarchies if individual associations or the path of classes are interesting to users. Although this kind of compression would achieve a more compact view for visualisations, it requires additional attention and investigation from users as it is applied on interesting parts of ontologies. There is also a cost of additional glyphs. In addition, as discussed in Chapter 1 multiple inheritance can exist in ontology hierarchies. It is currently addressed by duplicating classes under each of their parents, which introduces considerable additional data elements, especially when the duplicated classes are on shallow levels and are the roots of big subtrees. This kind of duplicated subtree could potentially also be compressed and replaced by single glyphs to reduce the size of

data elements in visualisations. However, this approach also requires extra investigation to understand the structure of these hidden subtrees.



(a) A hierarchy before further compression.

(b) The hierarchy after further compression.

Figure 7.1: Illustration of possible further compression for sibling leaf association classes (highlighted in the blue boxes) and *uninteresting* intermediate level classes (highlighted in the red boxes).

## 7.3 Future Directions

This research has addressed the research questions for visualising ontologies and associations, introduced in Chapter 1. However, there are many issues and challenges in the ontology visualisation research area. Several potential future works are outlined as follows.

### 7.3.1 Ontology Class Lables

During the discussions and user studies with the domain experts, one interesting finding is that they mostly focused on the class labels in visualisations. This might be a result that they get used to Protégé for their daily research, which is purely a list of class labels and represents ontologies in a highly expressive way. Although OntoPlot greedily displays the class labels where it is possible, there is room to improve. For example, to maintain the orientation of the association labels as horizontal, which is a more conventional reading direction than the diagonal orientation, one potential solution could be changing the orientation of the hierarchy instead. However, this may introduce new challenges to read the hierarchy structure and class labels aligned within ontology hierarchy. Thus, a smart and effective way to show the large number of ontology class labels in a compact visualisation requires more research.

### 7.3.2 Ontology Property Composition

One use case was discovered during the discussions with domain experts and is worth visualising. Ontology property composition, which links classes via different properties, constructing a chain of ontology associations, is important to visualise with the ontology hierarchy. One challenge of this work is how to define which classes should be included in this composition, as the association set may include all the classes in an ontology. One possible focus could be to dynamically identify and refine subsequent *interesting* classes based on user's current selection of classes.

### 7.3.3 Many-to-Many Ontology Association

Currently, the many-to-many associations between classes are shown as grid labels only for selected classes. It is beneficial to provide an overview of the association links between each pair of classes alongside the ontology hierarchy. One possibility is using a matrix view to do this as presented in Yang et al. (2016) and employing small mosaic box glyphs within the matrix to accommodate multiple properties. Another possibility is the use of hypergraphs to depict the associations between classes, which has been identified in Valdivia et al. (2021). More effective representations need to be explored to handle large ontologies and show the complex many-to-many mappings of associations in such ontologies.

### 7.3.4 Ontology Axioms

As discussed in Section 2.1, the associations defined in this research are expressed through the *someValuesFrom* restrictions. However, other data property restrictions like *allValuesFrom* and *hasValue*, which also contain rich information, are not handled by the current system. Moreover, besides the *subClassOf* axioms, there are more axioms involved in an ontology, such as the ones that express property *cardinality*, property value type *literals*, class *equivalence* and *disjointness*, and property inheritance hierarchy. Visualising these axioms is important for understanding the underlying structure of an ontology and the logical relationships between ontology classes and properties. Some visual annotations or visual languages have been developed to explore this topic, such as ODG (Silva-López et al., 2014) and VOWL (Lohmann, Negru, Haag and Ertl, 2014) that were mentioned in Chapter 3.2. These kind of visual representations work well for small subsets of axioms. How to increase their scalability to make sense for the whole ontology is still an open question.

### 7.3.5 Ontology Instances

Although most ontology visualisations focus on ontology classes, ontology *instances* that define individual objects for a class serving as fundamental components for a knowledge base, are also worth visualising. When talking about the instances that are contained within ontologies, these instance sets are often very large and treated as leaves in the ontology hierarchy, which substantially increases the width of the hierarchy and introduces new scalability issues to the visualisations, requiring more efficient use of space and more compact visual compression techniques. While coming to the instances that form knowledge graphs as backbones for linked data, the data volume grows to tens of millions (Paulheim, 2017). Whether visualising these instances alongside the ontology schema is useful to identify the relationships between entities and how to visualise such massive data is interesting to explore.

### 7.3.6 Multiple Ontologies

Visualising links between classes that are from more than one ontology is also an interesting topic. As discussed in Section 7.3.6, such a feature would facilitate important ontology

activities such as ontology mapping, ontology term reuse and ontology evolution investigation, but also face significant challenges to effectively accommodate the huge number of classes and associations contained in those ontologies.

The first question is how to align multiple ontology hierarchies together. For visualising two sets of data, here two ontologies, Ondov et al. (2018) defined five arrangements: stacked, adjacent, mirrored, overlaid and animated. When coming to more than two ontologies, the arrangement becomes more complicated. The technique term *small multiples* (Tufte, 1985) is used to depict this situation, where a series of similar representations using the same scale and axes are juxtaposed. Several approaches have been developed to visualise multiple hierarchies using the small multiples display (Chevenet et al., 2006; Telea and Auber, 2008; Burch and Lohmann, 2015), but are limited in terms of scalability. One can imagine that using visual compression techniques would allow a number of ontologies to be compactly visualised as a series of small graphics showing only the *interesting* parts among the ontologies.

Another question is how to show the links between multiple ontologies. Existing approaches used to visualise relationships between multiple representations can be categorised as following: edge drawing, colouring and matrix (Graham and Kennedy, 2010; Granitzer et al., 2010; Ivanova and Lambrix, 2014). While an individual ontology itself also contains associations between its own classes, how to effectively use these approaches to distinguish these intra-ontology associations with inter-ontologies relationships is worth further exploration.

## 7.4 Closing Remarks

This thesis has presented approaches for visualising large ontologies and ontology associations. These approaches have been shown to effectively support the exploration and analyses of complex ontology class relations. Their usability is demonstrated through user studies and case studies. This research not only advances the state-of-the-art in the ontology visualisation area but these techniques can also be applied to other hierarchically structured data visualisations, especially those showing non-hierarchical relations alongside hierarchy structures.



# Vita

Publications arising from this thesis include:

Yang, Y., Wybrow, M., Li, Y.-F., Czauderna, T. and He, Y. (2019). OntoPlot: A Novel Visualisation for Non-hierarchical Associations in Large Ontologies, *IEEE Transactions on Visualization and Computer Graphics* **26**(1): 1140–1150.

Yang, Y., Wybrow, M., Li, Y.-F., Czauderna, T. and He, Y. Exploring and visualising complex associations in biomedical ontologies with OntoPlot, **submitted to *Bioinformatics***.

Permanent Address: Faculty of Information Technology  
Monash University  
Australia

This thesis was typeset with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub><sup>1</sup> by the author.

---

<sup>1</sup>L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> is an extension of L<sup>A</sup>T<sub>E</sub>X. L<sup>A</sup>T<sub>E</sub>X is a collection of macros for T<sub>E</sub>X. T<sub>E</sub>X is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Glenn Maughan and modified by Dean Thompson and David Squire of Monash University.





# References

- Adam, D. (2002). The counting house, *Nature* **415**(6873): 726–729.
- Akrivi, K., Elena, T., Constantin, H., Georgios, L. and Costas, V. (2006). A Comparative Study of Four Ontology Visualization Techniques in Protégé: Experiment setup and preliminary results, *Tenth International Conference on Information Visualization, 2006. IV 2006.*, IEEE, pp. 417–423.
- Alani, H. (2003). TGVizTab: An Ontology Visualisation Extension for Protégé, *Knowledge Capture (K-Cap’03), Workshop on Visualization Information in Knowledge Engineering*, Sanibel Island, Florida, USA.
- Antoniou, G., Groth, P., Harmelen, F. v. v. and Hoekstra, R. (2012). *A Semantic Web Primer*, 3rd edn, The MIT Press.
- Archambault, D., Munzner, T. and Auber, D. (2008). GrouseFlocks: Steerable Exploration of Graph Hierarchy Space, *IEEE Transactions on Visualization and Computer Graphics* **14**(4): 900–913.
- Archambault, D., Purchase, H. C. and Pinaud, B. (2010). The Readability of Path-Preserving Clusterings of Graphs, *Computer Graphics Forum*, Vol. 29, Wiley Online Library, pp. 1173–1182.
- Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T. et al. (2000). Gene Ontology: tool for the unification of biology, *Nature Genetics* **25**(1): 25–29.
- Babaria, K. (2004). Using Treemaps to Visualize Gene Ontologies, *Human Computer Interaction Lab and Institute for Systems Research, University of Maryland, College Park, MD USA*.
- Bach, B., Pietriga, E., Liccardi, I. and Legostaev, G. (2011). OntoTrix: A Hybrid Visualization for Populated Ontologies, *Proceedings of the 20th International Conference Companion on World Wide Web*, pp. 177–180.
- Balzer, M. and Deussen, O. (2005). Voronoi Treemaps, *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, IEEE, pp. 49–56.
- Barton, A., Rosier, A., Burgun, A. and Ethier, J.-F. (2014). The Cardiovascular Disease Ontology, *FOIS*, pp. 409–414.
- Becker, R. A. and Cleveland, W. S. (1987). Brushing Scatterplots, *Technometrics* **29**(2): 127–142.
- Bruls, M., Huizing, K. and Van Wijk, J. J. (2000). Squarified Treemaps, *VisSym*, Springer, pp. 33–42.

- Buja, A., McDonald, J. A., Michalak, J. and Stuetzle, W. (1991). Interactive Data Visualization using Focusing and Linking, *IEEE Visualization*, Vol. 91, pp. 156–163.
- Burch, M., Konevtsova, N., Heinrich, J., Hoeflerlin, M. and Weiskopf, D. (2011). Evaluation of Traditional, Orthogonal, and Radial Tree Diagrams by an Eye Tracking Study, *IEEE Transactions on Visualization and Computer Graphics* **17**(12): 2440–2448.
- Burch, M. and Lohmann, S. (2015). Visualizing the Evolution of Ontologies: A Dynamic Graph Perspective, *VOILA@ ISWC*, p. 69.
- Card, S. K. and Nation, D. (2002). Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface, *Proceedings of the Working Conference on Advanced Visual Interfaces*, pp. 231–245.
- Chevenet, F., Brun, C., Bañuls, A.-L., Jacq, B. and Christen, R. (2006). TreeDyn: towards dynamic graphics and annotations for analyses of trees, *BMC Bioinformatics* **7**(1): 439.
- Cuenca-Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. and Sattler, U. (2008). OWL 2: The next step for OWL, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* **6**(4): 309–322.
- da Silva, I. C. S., Freitas, C. M. D. S. and Santucci, G. (2012). An Integrated Approach for Evaluating the Visualization of Intensional and Extensional Levels of Ontologies, *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors-Novel Evaluation Methods for Visualization*, ACM, p. 2.
- Dameron, O., Bettembourg, C. and Le Meur, N. (2013). Measuring the Evolution of Ontology Complexity: The Gene Ontology Case Study, *PLoS One* **8**(10): e75993.
- d’Aquin, M. and Noy, N. F. (2012). Where to publish and find ontologies? A survey of ontology libraries, *Journal of Web Semantics* **11**: 96–111.
- Demian, P. and Fruchter, R. (2006). Finding and understanding reusable designs from large hierarchical repositories, *Information Visualization* **5**(1): 28–46.
- Drummond, N., Horridge, M., Stevens, R., Wroe, C. and Sampaio, S. (2007). Pizza ontology, *The University of Manchester*.
- Drury, B., Fernandes, R., Moura, M.-F. and de Andrade Lopes, A. (2019). A survey of semantic web technology for agriculture, *Information Processing in Agriculture* **6**(4): 487–501.
- Dudáš, M., Lohmann, S., Svátek, V. and Pavlov, D. (2018). Ontology visualization methods and tools: a survey of the state of the art, *The Knowledge Engineering Review* **33**: e10.1–39.
- Dunne, C. and Shneiderman, B. (2013). Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, pp. 3247–3256.
- Dutkowski, J., Ono, K., Kramer, M., Yu, M., Pratt, D., Demchak, B. and Ideker, T. (2013). NeXO Web: the NeXO ontology database and visualization platform, *Nucleic Acids Research* **42**(D1): D1269–D1274.
- Dwyer, T., Koren, Y. and Marriott, K. (2006). IPSep-CoLa: An Incremental Procedure for Separation Constraint Layout of Graphs, *IEEE Transactions on Visualization and Computer Graphics* **12**(5): 821–828.

- Dwyer, T., Marriott, K. and Wybrow, M. (2008). Dunnart: A Constraint-Based Network Diagram Authoring Tool, *Graph Drawing*, Vol. 5417, Springer, pp. 420–431.
- Falconer, S. M. and Storey, M.-A. (2007). A Cognitive Support Framework for Ontology Mapping, *The Semantic Web*, Springer, pp. 114–127.
- Field, A., Miles, J. and Field, Z. (2012). *Discovering Statistics Using R*, SAGE Publications.
- Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D. and Antoniou, G. (2008). Ontology change: classification and survey, *The Knowledge Engineering Review* **23**(2): 117–152.
- Fraser, J., Adams, N., Macintosh, A., McKay-Hubbard, A., Lobo, T. P., Pardo, P. F., Martínez, R. C. and Vallecillo, J. S. (2003). Knowledge Management Applied to E-Government Services: The Use of an Ontology, *IFIP International Working Conference on Knowledge Management in Electronic Government*, Springer, pp. 116–126.
- Fruchterman, T. M. and Reingold, E. M. (1991). Graph Drawing by Force-directed Placement, *Software: Practice and Experience* **21**(11): 1129–1164.
- Fu, B., Noy, N. F. and Storey, M.-A. (2013). Indented Tree or Graph? A Usability Study of Ontology Visualization Techniques in the Context of Class Mapping Evaluation, *International Semantic Web Conference*, Springer, pp. 117–134.
- Fuchs, J., Isenberg, P., Bezerianos, A., Fischer, F. and Bertini, E. (2014). The Influence of Contour on Similarity Perception of Star Glyphs, *IEEE Transactions on Visualization and Computer Graphics* **20**(12): 2251–2260.
- Furnas, G. W. (1986). Generalized Fisheye Views, *ACM SIGCHI Bulletin* **17**(4): 16–23.
- Goodwin, S., Dykes, J., Jones, S., Dillingham, I., Dove, G., Duffy, A., Kachkaev, A., Slingsby, A. and Wood, J. (2013). Creative User-Centered Visualization Design for Energy Analysts and Modelers, *IEEE Transactions on Visualization and Computer Graphics* **19**(12): 2516–2525.
- Grabska, E. (1994). Graphs and Designing, *Graph Transformations in Computer Science*, Springer, pp. 188–202.
- Graham, M. and Kennedy, J. (2007). Visual exploration of alternative taxonomies through concepts, *Ecological Informatics* **2**(3): 248–261.
- Graham, M. and Kennedy, J. (2010). A survey of multiple tree visualisation, *Information Visualization* **9**(4): 235–252.
- Granitzer, M., Sabol, V., Onn, K. W., Lukose, D. and Tochtermann, K. (2010). Ontology Alignment — A Survey with Focus on Visually Supported Semi-Automatic Techniques, *Future Internet* **2**(3): 238–258.
- Gruber, T. R. et al. (1993). A translation approach to portable ontology specifications, *Knowledge Acquisition* **5**(2): 199–221.
- Guo, A., Racz, R., Hur, J., Lin, Y., Xiang, Z., Zhao, L., Rinder, J., Jiang, G., Zhu, Q. and He, Y. (2016). Ontology-based collection, representation and analysis of drug-associated neuropathy adverse events, *Journal of Biomedical Semantics* **7**(1): 29.

- He, Y., Sarntivijai, S., Lin, Y., Xiang, Z., Guo, A., Zhang, S., Jagannathan, D., Toldo, L., Tao, C. and Smith, B. (2014). OAE: The Ontology of Adverse Events, *Journal of Biomedical Semantics* **5**(1): 29.
- He, Y., Wang, H., Zheng, J., Beiting, D. P., Masci, A. M., Yu, H., Liu, K., Wu, J., Curtis, J. L., Smith, B. et al. (2019). OHMI: the ontology of host-microbiome interactions, *Journal of Biomedical Semantics* **10**(1): 1–14.
- Heer, J. and Card, S. K. (2004). DOITrees Revisited: Scalable, Space-constrained Visualization of Hierarchical Data, *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '04*, ACM, New York, NY, USA, pp. 421–424.
- Henry, N., Fekete, J.-D. and McGuffin, M. J. (2007). Nodetrix: a Hybrid Visualization of Social Networks, *IEEE Transactions on Visualization and Computer Graphics* **13**(6): 1302–1309.
- Herman, I., Melançon, G. and Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey, *IEEE Transactions on Visualization and Computer Graphics* **6**(1): 24–43.
- Hill, D. P., Smith, B., McAndrews-Hill, M. S. and Blake, J. A. (2008). Gene Ontology annotations: what they mean and where they come from, *BMC Bioinformatics* **9**(5): S2.
- Holten, D. (2006). Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, *IEEE Transactions on Visualization and Computer Graphics* **12**(5): 741–748.
- Horridge, M. (2005). OWLViz, [protegewiki.stanford.edu/wiki/OWLViz](http://protegewiki.stanford.edu/wiki/OWLViz), [github.com/protegeproject/owlviz](https://github.com/protegeproject/owlviz). Accessed June 2020.
- Horridge, M. and Bechhofer, S. (2011). The OWL API: A Java API for OWL Ontologies, *Semantic Web* **2**(1): 11–21.
- Horrocks, I., Patel-Schneider, P. F. and van Harmelen, F. (2003). From SHIQ and RDF to OWL: The Making of a Web Ontology Language, *Journal of Web Semantics* **1**(1): 7–26.
- Ivanova, V. and Lambrix, P. (2014). User Involvement for Large-Scale Ontology Alignment, *VISUAL@ EKAW*, pp. 34–47.
- Jankun-Kelly, T., Dwyer, T., Holten, D., Hurter, C., Nöllenburg, M., Weaver, C. and Xu, K. (2014). Scalability Considerations for Multivariate Graph Visualization, *Multivariate Network Visualization*, Springer, pp. 207–235.
- Jia, M., Li, L., Boggess, E., Wurtele, E. S. and Dickerson, J. A. (2010). Visualizing Multivariate Hierarchic Data Using Enhanced Radial Space-Filling Layout, *International Symposium on Visual Computing*, Springer, pp. 350–360.
- Jiao, Z. L., Liu, Q., Li, Y.-F., Marriott, K., Wybrow, M. et al. (2013). Visualization of large ontologies with landmarks., *GRAPP/IVAPP*, pp. 461–470.
- Jonquet, C., Toulet, A., Arnaud, E., Aubin, S., Yeumo, E. D., Emonet, V., Graybeal, J., Laporte, M.-A., Musen, M. A., Pesce, V. et al. (2018). AgroPortal: A vocabulary and ontology repository for agronomy, *Computers and Electronics in Agriculture* **144**: 126–143.
- Jurcik, M. A. (2012). Knoocks - ontology visualization plug-in for protégé, *Proceeding of CESCOG 2012: The 16th Central European Seminar on Computer Graphics*.

- Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology Mapping: The State of the Art, *The Knowledge Engineering Review* **18**(1): 1–31.
- Kamdar, M. R., Tudorache, T. and Musen, M. A. (2015). Investigating term reuse and overlap in biomedical ontologies., *ICBO* **2015**: 42–46.
- Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C. and Giannopoulou, E. (2007). Ontology Visualization Methods — A Survey, *ACM Computing Surveys (CSUR)* **39**(4): 10.
- Kruskal, J. B. and Landwehr, J. M. (1983). Icicle Plots: Better Displays for Hierarchical Clustering, *The American Statistician* **37**(2): 162–168.
- Kuhar, S. and Podgorelec, V. (2012). Ontology Visualization for Domain Experts: a New Solution, *2012 16th International Conference on Information Visualisation*, IEEE, pp. 363–369.
- Lam, H., Bertini, E., Isenberg, P., Plaisant, C. and Carpendale, S. (2011). Empirical Studies in Information Visualization: Seven Scenarios, *IEEE Transactions on Visualization and Computer Graphics* **18**(9): 1520–1536.
- Lanzenberger, M., Sampson, J. and Rester, M. (2010). Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data, *J. UCS* **16**(7): 1036–1054.
- Lin, C.-C. and Yen, H.-C. (2007). On Balloon Drawings of Rooted Trees, *Journal of Graph Algorithms and Applications* **11**(2): 431–452.
- Liu, Q., Wang, J., Zhu, Y. and He, Y. (2017). Ontology-based systematic representation and analysis of traditional Chinese drugs against rheumatism, *BMC Systems Biology* **11**(7): 130.
- Lohmann, S., Link, V., Marbach, E. and Negru, S. (2014). WebVOWL: Web-based visualization of ontologies, *International Conference on Knowledge Engineering and Knowledge Management*, Springer, pp. 154–158.
- Lohmann, S., Negru, S., Haag, F. and Ertl, T. (2014). VOWL 2: User-Oriented Visualization of Ontologies, *International Conference on Knowledge Engineering and Knowledge Management*, Springer, pp. 266–281.
- Lü, H. and Fogarty, J. (2008). Cascaded Treemaps: Examining the Visibility and Stability of Structure in Treemaps, *Proceedings of Graphics Interface 2008*, Canadian Information Processing Society, pp. 259–266.
- McGuffin, M. J. and Robert, J.-M. (2010). Quantifying the space-efficiency of 2D graphical representations of trees, *Information Visualization* **9**(2): 115–140.
- Moed, H. F. (2006). *Citation Analysis in Research Evaluation*, Vol. 9, Springer Science & Business Media.
- Motta, E., Mulholland, P., Peroni, S., d’Aquin, M., Gomez-Perez, J. M., Mendez, V. and Zablith, F. (2011). A Novel Approach to Visualizing and Navigating Ontologies, *International Semantic Web Conference*, Springer, pp. 470–486.
- Motta, E., Peroni, S., Gómez-Pérez, J. M., d’Aquin, M. and Li, N. (2012). Visualizing and Navigating Ontologies with KC-Viz, *Ontology Engineering in a Networked World*, Springer, pp. 343–362.

- Munzner, T. (2014). *Visualization Analysis and Design*, CRC Press.
- Neumann, P., Schlechtweg, S., Carpendale, S. et al. (2005). ArcTrees: Visualizing Relations in Hierarchical Data, *EuroVis*, pp. 53–60.
- Niroumand, H., Zain, M. F. M. and Jamil, M. (2013). Statistical Methods for Comparison of Data Sets of Construction Methods and Building Evaluation, *Procedia-Social and Behavioral Sciences* **89**: 218–221.
- Nobre, C., Gehlenborg, N., Coon, H. and Lex, A. (2018). Lineage: Visualizing Multivariate Clinical Data in Genealogy Graphs, *IEEE Transactions on Visualization and Computer Graphics* **25**(3): 1543–1558.
- Norman, D. A. (1988). *The Psychology of Everyday Things*, Basic books.
- Noy, N. F., McGuinness, D. L. et al. (2001). Ontology Development 101: A Guide to Creating Your First Ontology, *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, Stanford, CA.
- Noy, N. F., Shah, N. H., Whetzel, P. L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G. et al. (2009). BioPortal: ontologies and integrated data resources at the click of a mouse, *Nucleic Acids Research* **37**(suppl.2): W170–W173.
- Noy, N., Ferguson, R. and Musen, M. (2000). The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility, *Knowledge Engineering and Knowledge Management Methods, Models, and Tools* pp. 69–82.
- Ondov, B., Jardine, N., Elmqvist, N. and Franconeri, S. (2018). Face to Face: Evaluating Visual Comparison, *IEEE Transactions on Visualization and Computer Graphics* **25**(1): 861–871.
- Paulheim, H. (2017). Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods, *Semantic Web* **8**(3): 489–508.
- Peroni, S. and Shotton, D. (2018). The SPAR Ontologies, *International Semantic Web Conference*, Springer, pp. 119–136.
- Plaisant, C., Grosjean, J. and Bederson, B. B. (2002). SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation, *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, IEEE, pp. 57–64.
- Prud, E., Seaborne, A. et al. (2008). SPARQL Query Language for RDF, W3C recommendation, <http://www.w3.org/TR/rdf-sparql-query/>. Accessed June 2020.
- Reingold, E. M. and Tilford, J. S. (1981). Tidier Drawings of Trees, *IEEE Transactions on Software Engineering* **7**(2): 223–228.
- Rhee, S. Y., Wood, V., Dolinski, K. and Draghici, S. (2008). Use and misuse of the gene ontology annotations, *Nature Reviews Genetics* **9**(7): 509–515.
- Saghafi, A. (2016). Visualizing Ontologies - A Literature Survey, *International Conference on Conceptual Structures*, Springer, pp. 204–221.
- Salvadores, M., Alexander, P. R., Musen, M. A. and Noy, N. F. (2013). BioPortal as a Dataset of Linked Biomedical Ontologies and Terminologies in RDF, *Semantic Web* **4**(3): 277–284.

- Schulz, H.-J., Hadlak, S. and Schumann, H. (2011). The Design Space of Implicit Hierarchy Visualization: A Survey, *IEEE Transactions on Visualization and Computer Graphics* **17**(4): 393–411.
- Sedlmair, M., Meyer, M. and Munzner, T. (2012). Design Study Methodology: Reflections from the Trenches and the Stacks, *IEEE Transactions on Visualization and Computer Graphics* **18**(12): 2431–2440.
- Shah, N. and Musen, M. (2009). Ontologies for Formal Representation of Biological Systems, *Handbook on Ontologies*, Springer, pp. 445–461.
- Shneiderman, B. (1992). Tree Visualization with Tree-Maps: A 2-d Space-Filling Approach, *ACM Transactions on Graphics (TOG)* **11**(1): 92–99.
- Shneiderman, B. and Dunne, C. (2012). Interactive Network Exploration to Derive Insights: Filtering, Clustering, Grouping, and Simplification, *International Symposium on Graph Drawing*, Springer, pp. 2–18.
- Silva-López, R. B., Silva-López, M., Méndez-Gurrola, I. I. and Bravo, M. (2014). Onto Design Graphics (ODG): a graphical notation to standardize ontology design, *Mexican International Conference on Artificial Intelligence*, Springer, pp. 443–452.
- Sintek, M. (2003). OntoViz Tab: Visualizing Protégé Ontologies, <http://smi-protege.stanford.edu/svn/ontoviz-tab/>. Accessed June 2020.
- Smith, S. R., Barnard, D. T. and Macleod, I. A. (1984). Holophrasted displays in an interactive environment, *International Journal of Man-Machine Studies* **20**(4): 343–355.
- Staab, S. and Studer, R. (2009). *Handbook on Ontologies*, 2nd edn, Springer Publishing Company, Incorporated.
- Stasko, J. and Zhang, E. (2000). Focus + Context Display and Navigation Techniques for Enhancing Radial, Space-filling Hierarchy Visualizations, *IEEE Symposium on Information Visualization, 2000. InfoVis 2000.*, IEEE, pp. 57–65.
- Stearns, M. Q., Price, C., Spackman, K. A. and Wang, A. Y. (2001). SNOMED clinical terms: overview of the development process and project status., *Proceedings of the AMIA Symposium*, American Medical Informatics Association, p. 662.
- Stevens, R. and Lord, P. (2009). Application of Ontologies in Bioinformatics, *Handbook on Ontologies*, Springer, pp. 735–756.
- Storey, M.-A., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R. and Noy, N. (2001). Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protégé, *Workshop on Interactive Tools for Knowledge Capture*, Vol. 73.
- Storey, M.-A., Wong, K., Fracchia, F. D. and Muller, H. A. (1997). On Integrating Visualization Techniques for Effective Software Exploration, *IEEE Symposium on Information Visualization, 1997. Proceedings.*, IEEE, pp. 38–45.
- Takeda, H., Veerkamp, P. and Yoshikawa, H. (1990). Modeling Design Process, *AI Magazine* **11**(4): 37–48.
- Telea, A. and Auber, D. (2008). Code Flows: Visualizing Structural Evolution of Source Code, *Computer Graphics Forum*, Vol. 27, Wiley Online Library, pp. 831–838.
- Tufte, E. R. (1985). The Visual Display of Quantitative Information, *The Journal for Healthcare Quality (JHQ)* **7**(3): 15.

- Valdivia, P., Buono, P., Plaisant, C., Dufournaud, N. and Fekete, J.-D. (2021). Analyzing Dynamic Hypergraphs with Parallel Aggregated Ordered Hypergraph Visualization, *IEEE Transactions on Visualization and Computer Graphics* **27**(1): 1–13.
- Wagner, C., Cheung, K. S., Ip, R. K. and Bottcher, S. (2006). Building Semantic Webs for e-government with Wiki technology, *Electronic Government* **3**(1): 36–55.
- Wagner, J., Chelaru, F., Kancherla, J., Paulson, J. N., Zhang, A., Felix, V., Mahurkar, A., Elmqvist, N. and Corrada Bravo, H. (2018). Metaviz: interactive statistical and visual analysis of metagenomic data, *Nucleic Acids Research* **46**(6): 2777–2787.
- Wang, L., Li, M., Xie, J., Cao, Y., Liu, H. and He, Y. (2017). Ontology-based systematical representation and drug class effect analysis of package insert-reported adverse events associated with cardiovascular drugs used in China, *Scientific Reports* **7**(1): 1–14.
- Wang, T. D. and Parsia, B. (2006). CropCircles: Topology Sensitive Visualization of OWL Class Hierarchies, *International Semantic Web Conference*, Springer, pp. 695–708.
- Wang, W., Wang, H., Dai, G. and Wang, H. (2006). Visualization of Large Hierarchical Data by Circle Packing, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, pp. 517–520.
- Ward, M. O. (2002). A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization, *Information Visualization* **1**(3–4): 194–210.
- Ware, C. (2012). *Information Visualization: perception for design*, Elsevier, USA.
- Williamson, K. and Johanson, G. (2017). *Research Methods: Information, Systems, and Contexts*, Chandos Publishing.
- Wybrow, M., Elmqvist, N., Fekete, J.-D., Von Landesberger, T., van Wijk, J. J. and Zimmer, B. (2014). Interaction in the Visualization of Multivariate Networks, *Multivariate Network Visualization*, Springer, pp. 97–125.
- Wybrow, M., Marriott, K. and Stuckey, P. J. (2009). Orthogonal Connector Routing, *Graph Drawing*, Vol. 5849, Springer, pp. 219–231.
- Yang, Y., Dwyer, T., Goodwin, S. and Marriott, K. (2016). Many-to-Many Geographically-Embedded Flow Visualisation: An Evaluation, *IEEE Transactions on Visualization and Computer Graphics* **23**(1): 411–420.
- Yang, Y., Wybrow, M., Li, Y.-F., Czauderna, T. and He, Y. (2019). OntoPlot: A Novel Visualisation for Non-hierarchical Associations in Large Ontologies, *IEEE Transactions on Visualization and Computer Graphics* **26**(1): 1140–1150.
- Zhao, S., McGuffin, M. J. and Chignell, M. H. (2005). Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams, *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, IEEE, pp. 57–64.