Protégé-TS: An OWL Ontology Term Selection Tool

Ian Hyland and Renate A. Schmidt

The University of Manchester, UK

Abstract. This paper introduces Protégé-Term Selection (Protégé-TS), a software tool developed to support and partially automate the process of constructing OWL ontology signatures. Protégé-TS works from an OWL 2 based source ontology to assist the user in creating a signature which is composed of a list of concepts and roles. Such signature lists are often needed when working with ontologies. For example, the signature can be fed to a forgetting or modularisation tool in order to compute an output ontology which retains all the relevant logical entailments of the source ontology. The Protégé-TS tool implements a range of operations which allow the user to create and edit signatures of concept and role names in ontologies. This paper describes the functionality and architecture of the tool which has been implemented as a Java plugin to Protégé, and an evaluation of the tool when applied to several large scale ontologies, with an example focus on the medical ontology SNOMED-CT.

Keywords: OWL Ontology · Term Selection Approaches · Term Selection Tool · Protégé Plugin

1 Introduction

As a formal mechanism for knowledge representation, a wide range of ontologies have been constructed covering several application domains. Many of these ontologies have grown to such a size and complexity that their utility is undermined from the human perspective, for example, in understanding, performing maintenance and knowledge sharing, and for the ability of software to compute formal reasoning tasks.

An area of much recent research is in what is termed forgetting whereby a large ontology is reduced in size, and the retained portion of the ontology is focused on support of a given application domain use case. Several forgetting algorithms and software tools (e.g., FAME [36] and LETHE [22]) have already been developed, and can process a source ontology to generate a smaller output ontology which still retains all the relevant logical entailments of the source. These tools are fed with a signature of terms, i.e., concepts (OWL classes) and roles (OWL object properties), which specify the parts of the source ontology that are to be forgotten or kept.

The essence of this paper is the development of operations and strategies for creating term lists that constitute the signature, and the implementation of a software tool to support and partially automate generation of the signature for input into a forgetting tool or other purposes. Building upon the industry standard Protégé OWL ontology editor [30] with reasoning tool support, the term selection tool is implemented as a Protégé Java plug-in. The usability, performance, scalability and functionality of the tool is demonstrated to provide valuable support for execution, storing and replaying of term selection operations, and specifically is tested and evaluated against several large-scale ontologies, with a focus on SNOMED-CT [19].

2 Term Selection

Many different approaches to term selection are mooted in the literature.

Modularisation [5, 8, 9] requires that the user specifies a *seed signature* as input entry to compute a module of an ontology. All entities related in some respect to the chosen entity will be included in the module. With **information removal**, parts of the ontology are selected for removal, resulting in a module without all the detail of the original ontology. **Abstraction by breadth** is hiding unnecessary knowledge, it occurs when some relational properties of entities are removed to provide a simpler view, hence the *breadth* of the ontology is reduced. **Abstraction by depth** is also hiding unnecessary knowledge and occurs when high-level concepts from the source ontology are kept, and lower-level concepts are removed; hence the *depth* of the ontology is reduced.

FASTR [20] describes a unification-based system that efficiently identifies technical terms and demonstrates the complexity of the data that motivated the fundamental design decisions. [26] presents **statistical term extraction**, a statistical approach to terminology extraction which is general to all languages, although including some language-specific parameters. The method is used for the automatic identification of terminology and is theoretically and computationally simple and disregards resources such as linguistic or ontological knowledge. [23] describes **compound terms**, an evaluation of compound terms extraction from a corpus of the domain of Paediatrics. Bigrams and trigrams were automatically extracted from a corpus composed of 283 texts using three different extraction methods.

[28] describes a **terminology-driven** framework that integrates several components: automatic term recognition; term variation handling; acronym acquisition; and automatic term discovery of similarities and clusters. [15] presents a **natural language processing (NLP) based automatic extraction** protocol for specialised corpus analysis using NLP tools to define semantic hierarchies of verbs. In combining semantic and syntactic analysis of results in the verb macro structure it illustrates the evolution of the meaning from more general to more specific verbs.

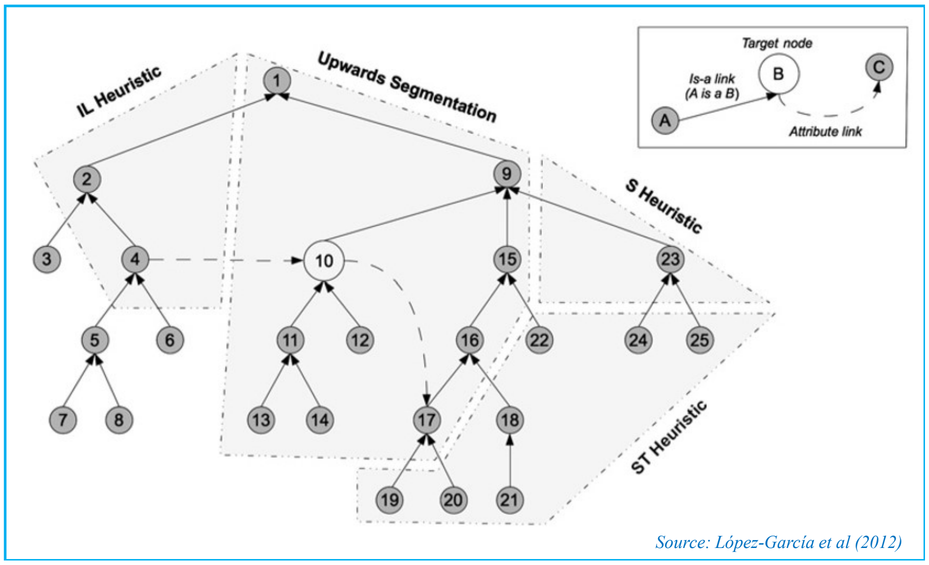


Fig. 1. Nodes identified by graph traversal heuristics.

Utilising **frequency-based filtering**, a study was conducted by [24] based on SNOMED-CT, whereby filtering of terms in MEDLINE [25] was used to reduce SNOMED-CT module sizes without discarding relevant concepts. Signature subsets were first extracted using four graph-traversal heuristics and one logic-based technique. These were subsequently filtered with frequency information from MEDLINE, Fig. 1 illustrates the four heuristics requiring identification of concept UpSets, DownSets and concept-role relationships.

Based on **graph-traversal and frequency data**, [24] summarises that graph-traversal strategies and frequency data drawn from an authoritative source can prune large ontologies and produce modules that exhibit acceptable coverage. However, it is also noted that evaluating the performance and optimality of modules is extremely difficult, and [5,7] concludes that "there is no universal way to modularize an ontology".

With **domain relevance concepts**, [34] notes that for effective search and management of large amounts of medical image and patient data, it is relevant to know the kind of information the clinicians and radiologists seek. They describe a statistical, clinical query pattern derivation as an approach to obtaining this information semi-automatically which is based on predicting clinical query patterns given medical ontologies, domain corpora and statistical analysis. Their aim was to discover radiologists' and clinicians' information needs by using semi-automatic text analysis methods that are independent of expert interviews. Additionally, [35] define a multi-perspective approach to term selection, which is based on three core assumptions. The goal is to reduce module size by using a more strictly selected and therefore more domain-specific set of ontology concepts.

The SNOMED-CT **Expression Constraint Language (ECL)** [18] is a formal syntax that enables the definition of a subset of SNOMED-CT concepts represented as an expression constraint. The expressions are computable rules used to define bounded sets of concepts gathered using constraint operations such as `descendantOf`, `ancestortOf` or `memberOf` and set-operations such as `AND`, `OR` and `MINUS`. The constraints can be used to restrict the selected concepts for a given concept, for example, as a machine-processable query, or a range of an attribute defined in the concept model. Two equivalent syntaxes are defined, a brief version used for machine to machine communication, and a long form which aids human readability. Several ECL browsers are publicly available including [4, 31,32], plus language parser tools such as SNOMED-CT Parser and ECL Parser.

Signature adjustment is used in an **iterative combined modularisation and forgetting** approach to ontology extraction [3]. This involves extension and if required partitioning of the given signature and evaluation by a domain expert. Based on a **framework for modularization**, [6] suggests an eight-stage framework to perform modularization of an ontology. The framework seems equally applicable to forgetting-based ontology processing.

The **forgetting tools** FAME [36] and LETHE [22] take as their input either a *forgetting signature*, i.e., a list of terms (concepts and roles) to be forgotten or a *keep signature*, i.e., a list of terms to be kept. Logically for a given ontology the sum of the forgetting and keep signatures must comprise all concepts and roles in the ontology. Both tools expect the user to provide the list of forget terms or keep terms. In the very basic GUI versions (of FAME and LETHE) the user can compose the relevant signature by selecting individual concepts/roles, or groups of concepts/roles, i.e., by using the *shift* and *control* keys in two windows listing all concept and role symbols occurring in the loaded ontology.

This approach is entirely usable for very small ontologies, where a **single shot** selection of the forgetting signature is adequate. Large ontologies like SNOMED-CT contain however hundreds of thousands of terms and scrolling through these to select individual terms is impractical. As an essentially single shot approach there is no support for an interactive and incremental approach to signature construction. The formula of the ontology is displayed, however all visibility of the ontology structure (concept and role hierarchies), axiom definitions associated with a given concept, any role domain and range restrictions, general class axioms, and annotations are not accessible. Each term is either selected, or it is not. No capability is present to more intelligently process terms based on their place in the hierarchy, e.g., UpSet, DownSet and Equivalence and other properties like General Class Axioms (GCAs) and role relationships between concepts. There is no search capability, e.g., to search for a given concept, role or annotation. There is no access to external reasoning capability, using tools like ELK [11] for classification, or the ability to apply queries along the lines of **DL Query** in Protégé. There is no capability to save and load a forgetting or keep signature, i.e., for archiving, sharing and off-line analysis. There is no help assistance, i.e., display of a help screen or pop-up help text. Clearly, the GUIs only support FAME or LETHE, and not for instance, other ontology extraction tools like the OWL API Module Extractor.

Because there are so many applications of term selection, and Protégé is a widely used ontology editor, we have developed a term selection plugin to give Protégé users the capability to conveniently and flexibly create term lists for use as input to forgetting or other tools, or for other purposes. The functional requirements of our Protégé-TS plugin were driven by the mentioned issues with the above term selection approaches.

3 Term Selection Tool Requirements

At the highest level the Protégé-TS tool supports the concurrent, interactive and incremental development of two lists of concepts and roles, termed the keep and forget signatures. High-level user interaction is illustrated in Fig. 2. The tool introduces a new **Term Selection** tab into the Protégé main menu. Upon clicking on any of the tab submenus a welcome screen is displayed, including pointers to help text and licence conditions. The welcome screen is displayed only once. A **Help Screen** is included to display more detailed instructions. Hovering the mouse pointer over a tab will show **pop-up help text**.

Various **error messages** are defined, for example, covering situations which require an ontology to have already been loaded `Error: First Load an Ontology` or an operation requiring that one concept or role is first selected `Error: Concept or Role Not Selected`. The **Metrics** tab shows basic metrics of the ontology and signatures, e.g., total numbers of concept, roles and axioms, and the concepts and roles that are assigned to the keep/forget signatures. Note that both keep and forget signatures are initially empty. The **Display** tab shows the current contents of the keep/forget signatures. A message will be displayed if the signature is too large to display on screen. The **Results** tab enables the displaying of the results of each operation performed to be toggled on/off.

The **All–Forget/Keep** tab will place all concepts and roles in the ontology into either the forget or the keep signature. When the command completes a summary of the ontology metrics and signatures is displayed. The main action of **All** is to update the concept and role annotation section. For example, running **All–Forget** (with reference to Fig. 3) the annotation for concept `A1` is updated to include `]-Term-Selection-Concept-Forget` and the role `r111` has been updated to include `]-Term-Selection-Role-Forget`, i.e., this indicates they are both in the forget signature. The **All–Clear** tab will set to empty all of the keep and forget signatures, i.e., this will delete the `]-Term-Selection-xxx` annotations.

The **Entity** tab adds a single entity (concept or role) to the keep/forget signatures. With this and the following actions, the individual concept/role annotation is updated with the appropriate text, e.g., `]-Term-Selection-Concept-Keep`. The **Equivalent** tab adds all equivalent concepts or roles to the keep/forget signatures. The **DownSet** tab adds the down set (i.e., all subconcepts or subroles) to the keep/forget signatures, and the **UpSet** tab adds the upset (i.e.,

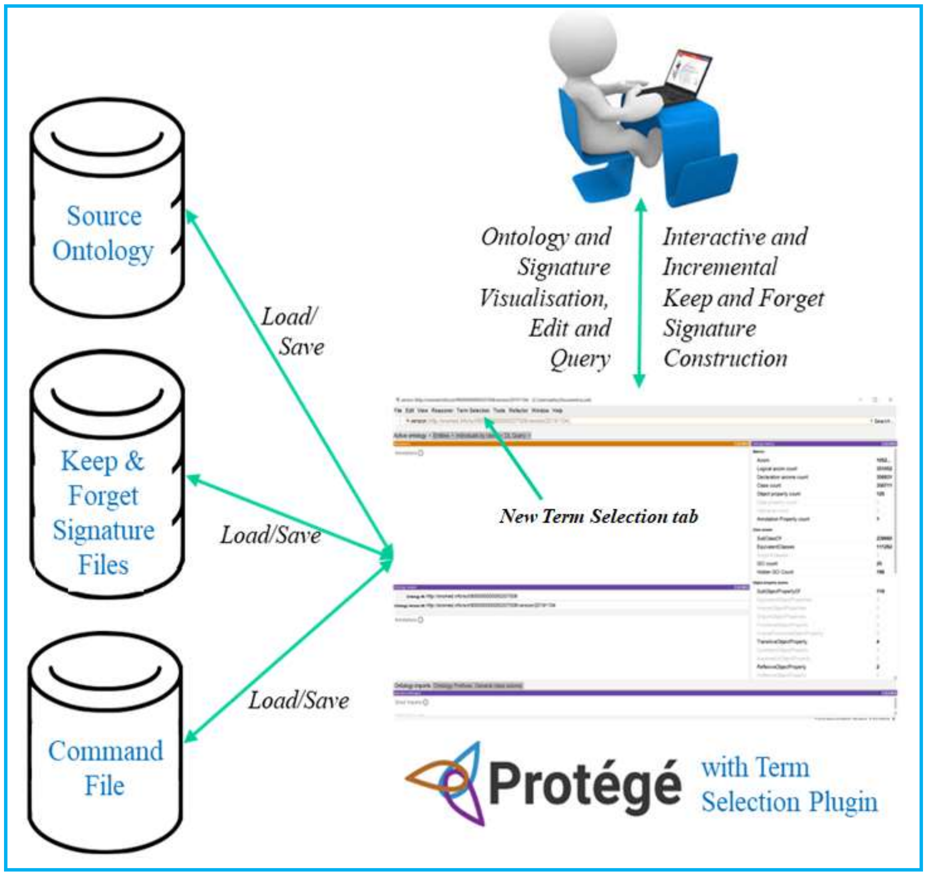

Source Ontology
Keep & Forget Signature Files
Command File
Load/ Save
Load/Save
Load/Save
Ontology and Signature Visualisation, Edit and Query
Interactive and Incremental Keep and Forget Signature Construction
New Term Selection tab
Protégé with Term Selection Plugin

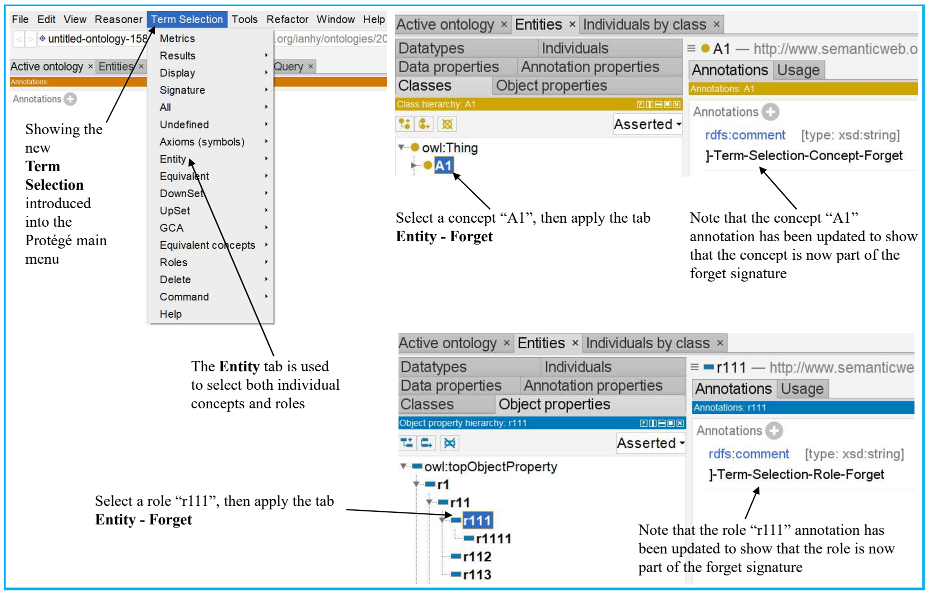
Fig. 2. Protégé Term Selection High-level User Interaction

all super concepts or super roles) to the keep/forget signatures. The **General Class Axioms (GCA)** tab adds to the forget/keep signatures all concepts and roles appearing in the GCA axiom definition. Note, the processing of these four operations utilises the reasoning capabilities as detailed below.

The **Roles** tab is applied to process **Concept-Role Chains**. Consider the example test ontology expressed in Manchester OWL Syntax (MOS)

```
T SubClassOf r1 some U        V SubClassOf r3 some W
U SubClassOf r2 some V        W SubClassOf r4 some X
U EquivalentTo EQ1            W EquivalentTo EQ2
```

which is shown diagrammatically in Fig. 4 (concepts in circles, roles as arrows). The user is prompted to enter the number of hops which is an integer in the range 1 to 8. The hop number controls how many of the concepts/roles are included in the processing to update the keep/forget signatures. For example,



Showing the new **Term Selection** introduced into the Protégé main menu

Select a concept “A1”, then apply the tab **Entity - Forget**

Note that the concept “A1” annotation has been updated to show that the concept is now part of the forget signature

The **Entity** tab is used to select both individual concepts and roles

Select a role “r111”, then apply the tab **Entity - Forget**

Note that the role “r111” annotation has been updated to show that the role is now part of the forget signature

Fig. 3. Protégé Term Selection Tab and Concept/Role Annotations

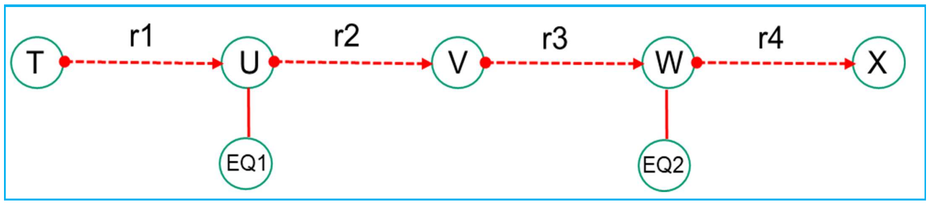


Fig. 4. Concept-Role Chains

selecting concept `T` and performing operation **Roles–1** includes concepts `T`, `U`, and role `r1` in the signature. Whereas performing **Roles–4** includes the concepts `T`, `U`, `V`, `W`, `X`, equivalent concepts `EQ1`, `EQ2` and roles `r1`, `r2`, `r3`, `r4`. Thus, concepts and roles encountered horizontally in four hops are included in the list.

Any given concept can be either *fully defined* or *primitive*. A fully defined concept is complete, i.e., it contains relationships that represent the full set of necessary and sufficient conditions [1]. In contrast, a primitive concept is incomplete, i.e., the set of conditions is insufficient to fully define the concept and therefore do not have any specified equivalent classes. By default, when processing the **Roles** function, both defined and primitive concepts are included in the signature. The **Equivalent concepts–Primitive only** tab turns-off inclusion of defined concepts, i.e., will exclude defined equivalent concepts and include only primitive concepts. Executing the **Roles–4** operation now excludes the equivalent classes `EQ1`, `EQ2` from the signatures.

When processing a concept, by default Protégé-TS will include all of the axiom definitions associated with the concept. For example, if a concept definition contains the axiom `r1 some Z` then both the `Z` concept and `r1` role are added to the signature. Protégé-TS includes the capability to toggle on/off the inclusion of axiom processing.

Any new concept or role added to an ontology will not automatically be part of the keep/forget signatures. The **Undefined** tab is used to set these undefined concepts and roles into either the keep or the forget signature. The **Delete** tab will delete from the ontology all concepts, roles and axioms definitions associated with the given concept that have been assigned to one of the keep or forget signatures. The delete capability was intended to be utilised in experiments with the OWL API Modularisation tool.

The **Signature Save/Load** tabs allow the keep and forget signatures to be saved to disk files or loaded from disk files. Note: only the filename needs to be entered, the .txt extension is added automatically, and error messages are displayed as appropriate. Protégé-TS will automatically keep an internal list of the commands as they are executed, with the ability to save **(Command–Save)** and replay the log as commands **(Command–Load)**. Operations that are applicable to the commands feature are **All**, **Delete**, **Undefined**, **Entity**, **DownSet**, **Upset**, **Equivalent**, **GCA**, **Roles**, **Results**, **Axioms** and **Equivalent Concepts**. The command log can also be cleared, using **Command–Clear**.

Considering **user deployment constraints**, Protégé-TS is intended to be utilised by research users without recourse to specialist hardware. The supported user run-time environment for Protégé-TS was therefore targeted to be a mid-range Microsoft Windows (version 10) laptop, or better. In performance and capacity terms, mid-range being defined as at least an Intel i5 CPU 4 cores, 8 GB memory and 132 GB disk.

From the perspective of **performance and scalability** the requirements were intended to support incremental and interactive development of the forget and keep signatures. To maintain a sleek user interface most of the functions detailed above should execute typically in a few seconds. The most “heavyweight” functions such as deleting large parts of the ontology, i.e., deleting concepts, roles and axioms identified by the keep or forget signature, were allowed to execute in less than a minute. Protégé-TS should support the larger available ontologies such as SNOMED-CT that can contain hundreds of thousands of concepts and axioms. Ontology expressivity should include support for all of OWL DL.

Concerning **licencing requirements**, Protégé-TS is available under the GNU General Public Licence version 3 or any later version. The welcome screen contains the standard text recommended in GNU [13]. Copies of the relevant Protégé-TS source files, e.g., Java code, build and configuration files, Test Cases, User Guide etc are publicly available on GitHub [16].

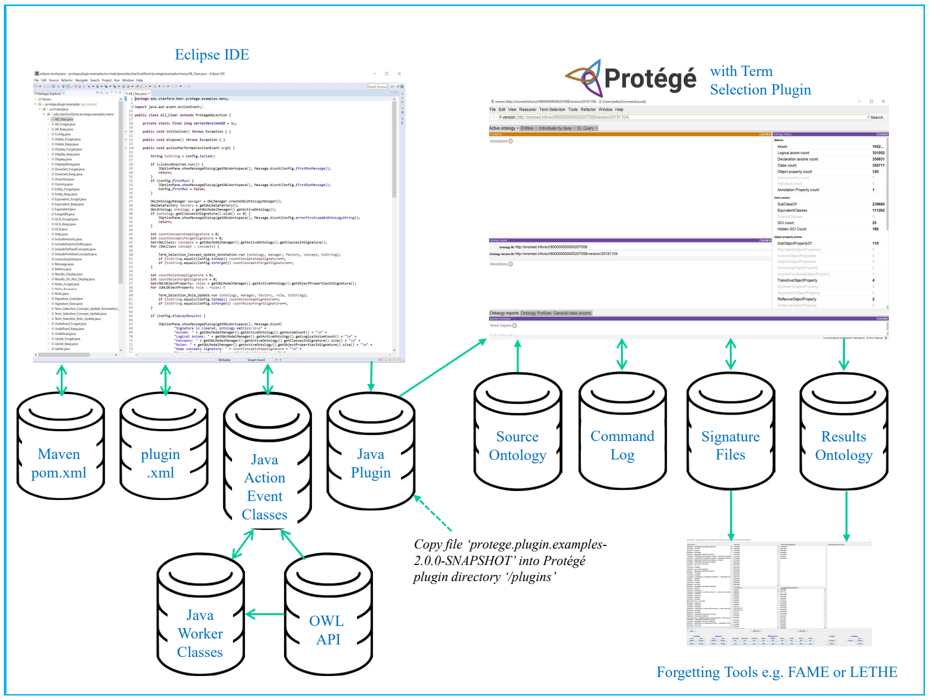

Eclipse IDE
Protégé with Term Selection Plugin
Maven pom.xml
plugin .xml
Java Action Event Classes
Java Plugin
Source Ontology
Command Log
Signature Files
Results Ontology
Java Worker Classes
OWL API
Copy file 'protege.plugin.examples-2.0.0-SNAPSHOT' into Protégé plugin directory '/plugins'
Forgetting Tools e.g. FAME or LETHE

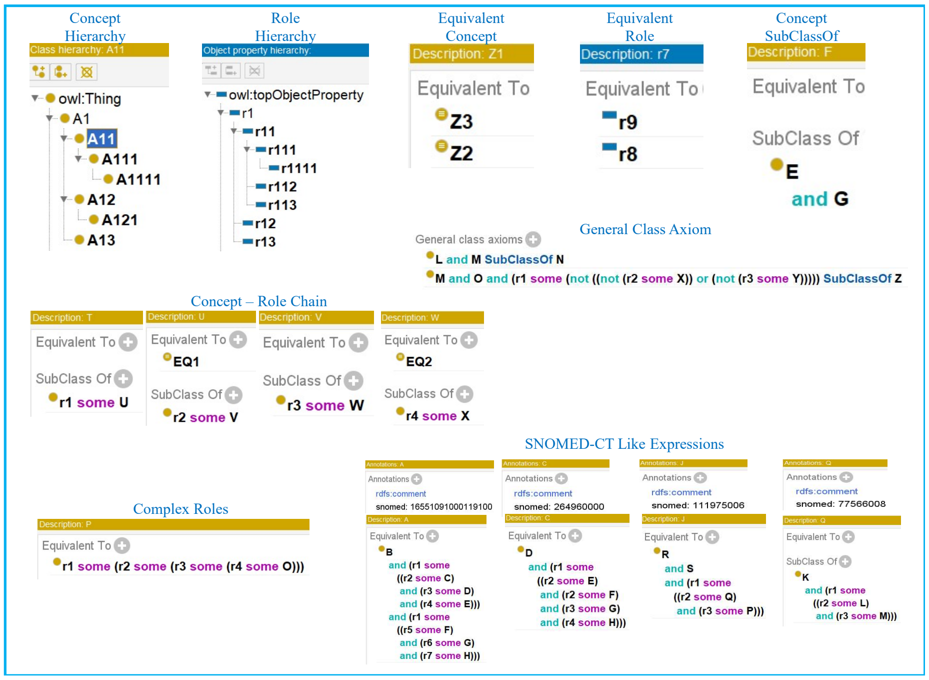
Fig. 5. Protégé-TS Software Architecture

4 Term Selection Tool Software Architecture

The software architecture of Protégé-TS is illustrated in Fig. 5 and has the following major components.

The **Eclipse Integrated Development Environment** [10] is used for editing of the Java [21] source and .xml files. The **Maven build system** specifies the build dependencies as detailed in the pom.xml file. The file specifies dependencies, i.e., groupId, artifactId and version number, to the Protégé editor and utilises Version 5.1.11 of the OWL API [29]. Additionally, it includes build instructions relating to the maven-compiler-plugin and maven-eclipse-plugin. A successful build generates the Protégé Java .jar plugin file. **Plugin.xml** is a file that specifies the layout of the new Protégé-TS tab, and the one-to-one association between each individual tab and a single Java Action Event Class.

The **Java Action Event Classes** are the code that is invoked when the mouse clicks on a given tab, these classes control all GUI actions. Each class has a one-to-one relationship with the entries in the plugin.xml file. The Java Action Event Classes utilise the **Java Swing library** to manage GUI actions, for example, displaying messages, number input and load/save filename. There are 34 of these classes in Protégé-TS. The **Java Worker Classes** provide support functions to the Java Action Event Classes, for example, updating concept and



Concept Hierarchy

Class hierarchy: A11

```
owl:Thing
  A1
    A11
      A111
        A1111
    A12
      A121
    A13
```

Role Hierarchy

Object property hierarchy:

```
owl:topObjectProperty
  r1
    r11
      r111
        r1111
      r112
      r113
    r12
    r13
```

Equivalent Concept

Description: Z1

Equivalent To
Z3
Z2

Equivalent Role

Description: r7

Equivalent To
r9
r8

Concept SubClassOf

Description: F

Equivalent To

SubClass Of
E
and G

General Class Axiom

General class axioms
L and M SubClassOf N
M and O and (r1 some (not ((not (r2 some X)) or (not (r3 some Y))))) SubClassOf Z

Concept – Role Chain

Description: T
Equivalent To
SubClass Of
r1 some U

Description: U
Equivalent To
EQ1
SubClass Of
r2 some V

Description: V
Equivalent To
SubClass Of
r3 some W

Description: W
Equivalent To
EQ2
SubClass Of
r4 some X

Complex Roles

Description: P
Equivalent To
r1 some (r2 some (r3 some (r4 some O)))

SNOMED-CT Like Expressions

Annotations: A
Annotations
rdfs:comment
snomed: 16551091000119100
Description: A
Equivalent To
```
B
 and (r1 some
   ((r2 some C)
    and (r3 some D)
    and (r4 some E)))
 and (r1 some
   ((r5 some F)
    and (r6 some G)
    and (r7 some H)))
```

Annotations: C
Annotations
rdfs:comment
snomed: 264960000
Description: C
Equivalent To
```
D
 and (r1 some
   ((r2 some E)
    and (r2 some F)
    and (r3 some G)
    and (r4 some H)))
```

Annotations: J
Annotations
rdfs:comment
snomed: 111975006
Description: J
Equivalent To
```
R
 and S
 and (r1 some
   ((r2 some Q)
    and (r3 some P)))
```

Annotations: Q
Annotations
rdfs:comment
snomed: 77566008
Description: Q
Equivalent To
SubClass Of
```
K
 and (r1 some
   ((r2 some L)
    and (r3 some M)))
```

Fig. 6. Test Ontology OWL Expression Types.

role annotations, traversing the ontology to process DownSet, UpSet, Equivalent concepts and GCA, construction of message content for display, save/load of signatures files and calculating ontology metrics. There are 22 of these classes in Protégé-TS.

The capabilities of the **OWL API** are used extensively by both the Java Action Event Classes and the Java Worker Classes. This includes both the inbuilt structural reasoner, supplemented by the ELK [11] reasoner which is optimised for processing ontologies with $\mathcal{EL}$ level expressivity.

Following execution of the Maven build, the plugin .jar file is generated and copied into the Protégé plugins directory. When Protégé is restarted, the plugin is loaded and the new Term Selection tab in the main menu is observed. Protégé-TS can then be utilised with the source ontology, command log and signature files to generate revised signature files and a results ontology containing the inserted term selection annotations ready to be processed by forgetting tools such as FAME and LETHE.

5 Evaluation of the Term Selection Tool

Functionality tests are performed against a test ontology containing a range of OWL expression types constructed to exercise the full set of functionalities described in Section 3. The test plan and results are available on GitHub [16]. Additional OWL expressions were defined that are based on the expression forms found in SNOMED-CT, see Fig. 6.

Detailed performance and scalability tests were executed against the SNOMED-CT ontology and are also available on GitHub [16]. All tests have passed, with one exception when deleting large parts of the ontology, the test exceeded the target allocated run-time limit. To test for the general application of Protégé-TS a small number of tests were executed against other sample ontologies, which encompass different expressivity and scale. Examples included FMA [12], GO [14], Uberon [33], ICD [17] and NCIt [27].

Usability testing was performed using the System Usability Scale (SUS) which was a simple 10 item questionnaire, designed by [2] in the mid-1980s to assess a user's overall satisfaction with a product. Three academic staff and research users, who are all highly experienced in OWL, Protégé and the SNOMED-CT ontology, performed the test, and the 88% SUS score achieved indicates a high-level of perceived usability.

Several use case questions were defined to exercise the available features of Protégé-TS and evaluate its real-world utility. Example questions were to find "cause for severe sunburn damaged skin", "available blood pressure measurement techniques", and "list of bones in the hand". Taking the first question and searching on "sunburn skin disorder" returned three results covering first, second and third-degree sunburn. For example, expressed in MOS:

```
'Sunburn of first degree (disorder)' EquivalentTo
'Acute effect of ultraviolet radiation on normal skin (disorder)'
and ('Role group (attribute)' some (('Associated morphology (attribute)' some
  'First degree burn injury (morphologic abnormality)')
and ('Causative agent (attribute)' some 'Ultraviolet radiation (physical force)')
and ('Finding site (attribute)' some 'Skin structure (body structure)')))
```

Expressed in natural language: "severe sunburn is a first-degree burn injury of normal skin caused by ultraviolet radiation". In Protégé selecting the concept `'Sunburn of first degree (disorder)'`, by utilising the features of Protégé-TS allows the keep/forget signatures to be generated and subsequently fed into FAME and LETHE. The features employed include **Entity**, **DownSet**, **UpSet**, **GCA** and **Roles-1** through to **Roles-8**. The number of concepts and roles added to the signatures as a result of each operation is shown in Table 1. For instance, **Entity - Keep** without axiom inclusion will add just the concept to the keep signature, whereas with axioms included will add the concept itself plus the 4 concepts and 4 roles that form the axiom definition. Inclusion of axiom processing increases run-time significantly from 2 to 9 seconds. **UpSet** w/ axioms generates a signature with 39 concepts and 7 roles, but the extensive OWL API computation involved leads to slightly excessive run-time.

Perhaps most closely related to Protégé-TS is SNOMED-CT's **Expression Constraint Language (ECL)** [1,18]. A high-level feature comparison of Protégé-TS versus ECL shows that ECL supports: UpSet and DownSet not including self, combined constraints, relationships and cardinality. Whereas Protégé-TS

Table 1. Test Results `'Sunburn of first degree (disorder)'`

Action	Run-Time (secs)	Concepts Added	Roles Added	Test Comment
Restart Protégé	17			Pass
Load SNOMED-CT	52			Pass
All-Forget	5	350711	120	Pass
Search `'sunburn skin analysis'`	1			Pass
Entity-Keep w/o Axioms	2	1	0	Pass
Entity-Keep w/ Axioms	9	5	4	Pass
DownSet w/ Axioms	21	5	4	Pass
UpSet w/ Axioms	49	39	7	Partial Pass
Equivalent-Keep w/ Axioms	17	5	4	Pass
GCA-Keep w/ Axioms	2	1	0	Pass
Role-Keep 1	7	5	4	Pass
Role-Keep 2	7	13	4	Pass
Role-Keep 3	7	24	4	Pass
Role-Keep 4	6	31	4	Pass
Role-Keep 5	6	33	4	Pass
Role-Keep 6	8	34	4	Pass
Role-Keep 7	6	34	4	Pass
Role-Keep 8	6	34	4	Pass

supports processing of equivalent concept and role definitions, general class axiom definitions, axiom definitions associated with a given concept, and concept-role chains. Protégé-TS also supports incremental interactive signature creation, editing, load/save. Lastly, of course ECL is dedicated to SNOMED-CT, whereas being available as a Protégé-module Protégé-TS can be applied to any OWL 2 based ontology and benefits from the various functionality available in Protégé.

6 Conclusion

Several term selection approaches have been identified and a tool has been implemented to support the core functions required by some of these approaches. Different term selection operations are available in the Protégé-TS plugin, from the simple case of selecting individual terms, through the more complex case of where the ontology structure (UpSet, DownSet, GCA and axiom definitions etc) is automatically computed to specify selected terms. Several even more complex approaches could form the basis for further work. Various quantitative and qualitative metrics were specified to test and evaluate the tool. The experimental results demonstrate the functionality, performance, scalability and general applicability of Protégé-TS, i.e., the tests cases pass (with minor exceptions of excessive run-time) and the tool is determined to have met its stated requirements. Validation of the tool encompassed performing a System Usability Scale Questionnaire, which provided evidence of a high level of usability and ideas for future developments.

References

1. Bhattacharyya, S.B.: Introduction to SNOMED-CT. Springer (2016)
2. Brooke, J.: SUS: A "quick and dirty" usability scale. In: Jordan, P., Thomas, B., Weerdmeester, B. (eds.) Usability Evaluation in Industry, pp. 189–194. Taylor & Francis (1996)
3. Chen, J., Alghamdi, G., Schmidt, R.A., Walther, D., Gao, Y.: Ontology extraction for large ontologies via modularity and forgetting. In: Kejriwal, M., Szekely, P.A., Troncy, R. (eds.) Proceedings of the 10th International Conference on Knowledge Capture (K-CAP'19). pp. 45–52. ACM (2019)
4. CSIRO: `https://apg.ihtsdotools.org/`. Accessed 4 November 2019.
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research **31**, 273–318 (2008)
6. d'Aquin, M.: Modularizing ontologies. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 213–33. Springer (2012)
7. d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M.: Criteria and evaluation for ontology modularization techniques. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, Lecture Notes in Computer Science, vol. 5445, pp. 67–89. Springer (2009)
8. Del Vescovo, C.: The modular structure of an ontology: Atomic decomposition towards applications. In: Proceedings of the 24th International Workshop on Description Logics (DL'11). CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
9. Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Winget, A.: Decomposition and modular structure of bioportal ontologies. In: The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Lecture Notes in Computer Science, vol. 7031, pp. 130–145. Springer (2011)
10. Eclipse: `https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers`. Downloaded 28 July 2019
11. ELK: `https://protegewiki.stanford.edu/wiki/ELK.VersionELK_0.4.3`. Accessed 3 October 2019
12. FMA: `https://bioportal.bioontology.org/ontologies/FMA`. Accessed 3 November 2019.
13. GNU: `https://www.gnu.org/licenses/gpl-3.0.en.html`. Accessed 12 November 2019
14. GOC: The Gene Ontology Consortium. Gene Ontology Annotations and Resources. Nucleic Acids Research, 41(D1):D530–D535, 2013
15. Goncharova, Y., Sánchez Cárdenas, B.: Specialized corpora processing with automatic extraction tool. Procedia: Social and Behavioral Sciences **95**, 293–297 (2013)
16. Hyland, I.: Protégé-TS (2019), `https://github.com/ianhyland/Protege-TS.git`. Accessed 23 May 2020.
17. ICD: `https://www.who.int/classifications/icd/en/`. Accessed 4 November 2019
18. IHTSDO: Expression constraint language: Specification and guide, `https://confluence.ihtsdotools.org/display/DOCECL/Expression+Constraint+Language+-+Specification+and+Guide`. Accessed 3 November 2019
19. IHTSDO: SNOMED-CT, `https://www.snomed.org/`. Accessed 17 July 2019
20. Jacquemin, C.: Spotting and Discovering Terms through Natural Language Processing. MIT Press (2001)

21. Java: Java Development Toolkit https://www.oracle.com/technetwork/java/javase/downloads/index.html, Version-10.0.1. Downloaded 12 July 2019
22. Koopmann, P.: Practical Uniform Interpolation for Expressive Description Logics. Ph.D. thesis, The University of Manchester, UK (2015)
23. Lopes, L., Vieira, R., José Finatto, M., Martins, D.: Extracting compound terms from domain corpora. Journal of the Brazilian Computer Society **16**, 247–259 (2010)
24. López García, P., Boeker, M., Illarramendi, A., Schulz, S.: Usability-driven pruning of large ontologies: The case of SNOMED-CT. J. Am. Med. Inform. Assoc. **19**(e1), e102–e109 (2012)
25. MEDLINE: https://www.nlm.nih.gov/bsd/pmresources.html. Accessed 3 November 2019
26. Nazar, R.: A statistical approach to term extraction. International Journal of Engineering and Science **11**(2), 159–182 (2011)
27. NCIT: https://ncit.nci.nih.gov/ncitbrowser/. Accessed 3 November 2019
28. Nenadic, G., Spasic, I., Ananiadou, S.: Terminology-driven mining of biomedical literature. Bioinformatics **19**(8), 939–943 (2003)
29. OWL-API: https://github.com/owlcs/owlapi. Version 5.1.11. Downloaded 12 July 2019
30. Protégé: http://protege.stanford.edu/. Accessed 6 July 2019
31. SNOW: https://mq.b2i.sg/snow-owl/#. Accessed 2 November 2019.
32. SNQuery: https://snquery.veratech.es/. Accessed 2 November 2019.
33. Uberon: https://bioportal.bioontology.org/ontologies/UBERON. Accessed 4 November 2019
34. Wennerberg, P., Buitelaar, P., Zillner, S.: Deriving clinical query patterns from medical corpora using domain ontologies. In: Workshop Biomedical Information Extraction 2009. pp. 50–56. Association for Computational Linguistics, USA (2009)
35. Wennerberg, P., Schulz, K., Buitelarr, P.: Ontology modularization to improve semantic medical image annotation. Journal of Biomedical Informatics **44**, 155–162 (2011)
36. Zhao, Y.: Automated Semantic Forgetting for Expressive Description Logics. Ph.D. thesis, The University of Manchester, UK (2018)