SWAT Editing Tool

Version 1.1

Getting started

The SWAT Editing Tool allows you to develop ontologies for the semantic web, using a controlled dialect of English rather than a computer language. The version described here uses an exceptionally simple dialect called OWL Simplified English, which covers the most commonly used OWL patterns. This document introduces the language in easy steps, and explains how to use the editor.

To create an ontology, you write a text in OWL Simplified English. You can either type this text into the editing pane of the tool, or load it from a text file produced with any text editor. Here is a short example of a text.

A dog is an animal.
A cat is an animal.
No dog is a cat.
Rover is a dog.
Kitty is a cat.

Each sentence in the text is interpreted as a single OWL statement. Sentences must end in a full stop followed by a new line character.

As can be seen, the text illustrates two kinds of statements, one generic and one specific. Generic statements are about *classes* such as dogs and cats; specific statements are about *individuals* such as Rover and Kitty. In the editing pane, individual and class names are highlighted using a colour code: individual names are shown in violet, class names in orange. (This colour code is also used in the Protégé graphical editor.)

To get started, try typing the above text into the editing pane (or transfer it from this document using copy and paste). As you type text into the editing pane, the program will provide various kinds of *feedback* for your guidance (in addition to the colour highlighting just mentioned). The feedback shows you (a) how the sentence has been interpreted so far as an OWL statement, and (b) the options for continuing the sentence. This information appears in two further panes called the *message pane* and the *options pane*, using the following layout.

<table>
<tr><td>EDITING PANE</td><td>OPTIONS PANE</td></tr>
<tr><td colspan="2">MESSAGE PANE</td></tr>
</table>

To save the ontology in one of the standard OWL formats, open the *File* menu and choose *Save XML*. An ontology corresponding to the text will be saved in OWL/XML format. In that form it can be loaded into a graphical tool such as Protégé.

Step 1

Simple statements about classes

The following text illustrates some statements describing relationships between classes. A class is a *type* of individual, such as a person or a city, as opposed to an individual such as Winston Churchill or London.

A motor bike is a motor vehicle. No car is a motor bike.

Two schematic patterns are shown here. The first states that one class is a subclass of another; the second states that one class is disjoint (i.e., entirely separate) from another. In the options pane these patterns are presented as follows:

A [class] is a [class]. No [class] is a [class].

To add such a statement to the text, you begin by placing the cursor at the start of a new line in the editing pane (hitting carriage return if necessary). The patterns just shown will then appear in the options pane. At this point, you can either click on a pattern, or simply start typing, whichever you find easier. If you click on the top pattern, the editing pane will look like this:

A motor bike is a motor vehicle. No car is a motor bike. A [class] is a [class].	

Selecting one of the place-holders will then yield options based on the class names entered so far:

A motor bike is a motor vehicle. No car is a motor bike. A [class] is a [class].	car motor bike motor vehicle

If you want one of these class names, click on it; if instead you need a new class name, simply type it in. The place-holder will be replaced by the text that you type.

One advantage of the SWAT editing tool is that there is no need to declare class names: you can simply start using them and the program will add them to

its list. However, to allow the program to detect phrases that are class names, you must conform to some simple rules. These rules do not guarantee that the name will be grammatical — this is left to you. They do guarantee that the program can work out where class names begin and end.

A class name is formed from any sequence of the following words: noun, adjective, proper name, preposition, definite article, number, or string. It must not contain a verb, or conjunction, or relative pronoun, or indefinite article.

A number can be either an integer or a decimal (e.g., 365, 3.14); a string is anything inside double-quotes (e.g., "Pride and Prejudice"). The following table gives examples.

yellow submarine	Correct
man in the moon	Correct, the definite article is allowed
Rolls Royce 1912	Correct, proper names and numbers are allowed
"Fish and Chips" shop	Correct, the conjunction is within a string
yellow of of of	Correct, although obviously ungrammatical
steak and kidney pie	Incorrect, contains conjunction (not within string)
building works manager	Incorrect only if 'works' is declared as verb
friend of a friend	Incorrect, contains indefinite article

In practice the main problem here is the ambiguity of present-tensed verbs with plural nouns, as in the example 'building works manager'. To resolve this ambiguity, all words used as verbs must be declared; we will describe later how this is done. If a word like 'works' is not declared as a verb, it will be interpreted as a noun and thus allowed within a class name.

Step 2

Simple statements about individuals

London is a city.
The mayor of London is a man.

Both these statements concern a specific individual — a place in one case, a person in the other. Logically, they both assert *class membership*: London belongs to the class *city* (which also includes for example Paris and Rome); the mayor of London belongs to the class *man*. Schematically this pattern is represented as follows in the options pane:

[Individual] is a [class].

A statement of this form can be added to the text as described in Step 1, by selecting the above pattern and replacing its place-holders by suitable individual or class names. Alternatively you can simply type in a sentence of the appropriate form.

To ensure that the sentence is interpreted correctly, you must adhere to the following rule for forming names of individuals.

An individual name must begin either with a proper name, or with the definite article 'the'. This may be followed by one or more of the words permitted in class names (i.e., noun, adjective, proper name, preposition, definite article, untensed verb, number, string). If the opening word is 'the', at least one such word must be added.

As with class names, you must take care not to include the indefinite article, conjunctions, relative pronouns, or tensed verbs, any of which will be interpreted as signals that the individual name has already ended. Note that proper names are distinguished by starting with a capital letter: thus 'Cat' will be interpreted as a proper name, and 'cat' as a noun. Examples:

Spider man	Correct
Bert "Bruiser" Higgins	Correct, strings are allowed
Eleanor of Aquitaine	Correct, prepositions are allowed
the man in the moon	Correct, the definite article is allowed
the 1921 Rolls Royce	Correct, numbers are allowed
Pride and Prejudice	Incorrect, contains conjunction
Poverty is Violence	Incorrect, contains tensed verb
spider man	Incorrect, does not open with 'the' or proper name

Step 3

Simple statements containing properties

A *property* in OWL is a relationship between two individuals. For instance, a relationship between Queen Elizabeth and Prince Charles is that Queen Elizabeth *is mother of* Prince Charles. Or a relationship between Queen Elizabeth and Buckingham Palace is that Queen Elizabeth *lives in* Buckingham Palace.

In OWL Simplified English, properties are named by phrases starting with a verb in the present tense. Thus the phrase 'is mother of' starts with 'is', while the phrase 'lives in' starts with 'lives'.

To distinguish property names from individual and class names, they are highlighted in blue (again following the convention used in Protégé).

Queen Elizabeth lives in Buckingham Palace.

Properties can also be used in statements containing classes, as in the following sentences:

Queen Elizabeth lives in a palace.
An English queen lives in Buckingham Palace.
A queen lives in a palace.

It is important to understand exactly how these sentences are interpreted.

- The first means that there is at least one palace (possibly more than one) that Queen Elizabeth lives in.
- The second means that every English queen, without exception, lives in Buckingham Palace. If you want to emphasize this you can write 'Every English queen lives in Buckingham palace', which in OWL Simplified English means exactly the same.
- The third means that for every queen without exception, there is at least one palace that this queen lives in. It might be the same palace, or a different one for each queen; this is not specified. Again, you can use the alternative wording 'Every queen lives in a palace' with exactly the same meaning.

Schematically these patterns are represented as follows in the options pane:

[Individual] [has-property] [Individual].
[Individual] [has-property] a [class].
A [class] [has-property] [Individual].
A [class] [has-property] a [class].

In a sentence like 'A queen lives in a palace', the editing tool must infer somehow that the word 'lives' opens a property name rather than continuing a class name. To make this possible, you must declare 'lives' as a verb, and use it *only* in this capacity. The declaration is added to the text using a special metadata statement starting with the symbol '#' and continuing with the key word VERB (lower case is also allowed), followed by the root and –s forms of the verb:

VERB live lives

Such statements may appear anywhere in the text. There is no need to declare the auxiliary verbs 'is' and 'has'.

The rule for forming property names is as follows.

A property name must begin with a verb in the present tense: either 'is' (or plural 'are') or 'has' ('have'), or a word declared as a verb. This may be followed by any sequence of the following words: noun, adjective, preposition, untensed verb. If the opening word is an auxiliary ('is' or 'has' or their plurals), at least one further word must be included.

Note that the words following the opening word must not include numbers, strings, or proper names. These word categories would signal the opening of a new name, such as an individual name (as in the phrase 'lives in Buckingham Palace'). Examples:

has as father	Correct
is friend of	Correct
is of of of	Correct, although ungrammatical
is owned by	Correct, past participle is allowed
comes from	Correct if 'comes' is declared as a verb
usually comes from	Incorrect, does not open with verb
has	Incorrect, lacks a continuation after the auxiliary
visits London with	Incorrect, contains proper name
has over 20 friends on	Incorrect, contains number
is owned and run by	Incorrect, contains conjunction

Step 4

Simple statements containing literals

A *literal* in OWL is a data value such as an integer. OWL provides for a wide range of datatypes, of which only three are covered in OWL Simplified English: integer, decimal and string. In statements, literals always occur as values of a *property*, as in the following statements:

The battle of Waterloo is dated in 1815.
Napoleon has surname "Bonaparte".

As can be seen, literals are highlighted in green.

OWL actually distinguishes between two kinds of property. An *object property* is a relationship between two individuals (as described in Step 3). A *data property* instead is a relationship between an individual and a literal. Thus in the examples above, the phrases 'is dated in' and 'has surname' are names of data properties, not object properties. The editor uses blue highlighting for both kinds of properties, since they are distinguished anyway by the colour of the phrase that follows (green for a data property, violet or orange for an object property).

The rule for forming literal names is as follows.

A literal must either be an integer, or a decimal, or a string. An integer is a series of digits (0–9). A decimal is a series of digits including an internal decimal point (full stop character). A string is a sequence of characters within double-quotes, provided these characters do not include new-line or double-quote.

Examples:

365	Correctly formed integer
36.5	Correctly formed decimal
.5	Incorrect, decimal point not internal
1914-1948	Incorrect, contains hyphen
"1914-1918"	Correctly formed string
"Pride and Prejudice"	Correctly formed string
'Pride and Prejudice'	Incorrect because single quotes are used
"The "X" generation"	Incorrect, contains an embedded string

Step 5

Statements with plural objects

So far we have considered only statements in which the object of a property name is singular:

A spouse is married to a person.

The interpretation of this sentence is that every spouse is married to at least one person. If you wish, you can make this absolutely clear as follows:

A spouse is married to at least one person.

Using this phrasing, we can obviously replace 'one' with other integers, allowing us for example to describe a bigamist:

A bigamist is married to at least two persons.

This is an example of a *numerical restriction*. Such restrictions require several additions to the language. First, we need integers such as 2 or 365; for convenience, the words one, two, etc. may be used for integers in the range 1-10. Second, we need phrases corresponding to the arithmetical relations '$\geq$', '$\leq$', '$=$' which qualify the integer; for this purpose we use 'at least', 'at most', and 'exactly'. Finally, for integers above 1, the class name should be put into the plural – 'persons' rather than 'person'.

Since the controlled language is not designed to enforce grammatically correct English, you can if you wish write 'at least two person' as well as 'at least one person'; this makes it obvious that the two names refer to the same class. If instead you write 'at least two persons', the program must work out whether 'person' and 'persons' refer to the same class, and to do this it applies the following rule:

To obtain the plural form of a class name, first identify the head word, which is either the first word preceding a preposition or, if there is no preposition, the final word of the class name. Then apply normal rules of English morphology to derive the –s form of the head word.

The program has a list of common irregular plurals which would allow it (for instance) to recognise 'people' as a plural of 'person'.

Here are some examples of applications of the plural rule.

Singular	**Plural**
yellow submarine	yellow submarines
house near the shops	houses near the shops
Rolls Royce 1912	Rolls Royce 1912s
young Mozart	young Mozarts
man from the shop near Boots	men from the shop near Boots

The full set of predicate patterns including plural forms is shown schematically below; these options (among others) would be displayed by the editing tool if the subject of the current sentence had already been specified (e.g., 'John' or 'Every person').

is a [class].
[has-data-property] [literal].
[has-property] [Individual].
[has-property] a [class].
[has-property] exactly [integer] [class].
[has-property] at least [integer] [class].
[has-property] at most [integer] [class].
[has-property] only [class].

The final pattern in this list allows statements restricting the value of the property to a specified class, as in this example:

Every woman is married to only men.

Note that this sentence allows that a woman may be unmarried, or married to more than one man; what it disallows is that a woman is married to something that is not a man (e.g., another woman, or a bottle of milk).

Step 6

Statements about properties

OWL provides for several kinds of statement about properties, of which the most common concern their *domain* and *range*. Domain statements constrain the class of individuals that can occur in the 'subject' role; range statements constrain the class of individuals that can occur in the 'object' role. Thus for the sentence pattern 'X lives in Y' we might want to stipulate that X must be a person and Y must be a place, or in other words that the domain of 'lives in' is the class *person* and the range is the class *place*.

Statements about properties are very rare in normal English (this kind of information is taken for granted). Perhaps because of this, it is hard to find sentence patterns that express them fluently. In OWL Simplified English the following patterns are used for domain and range.

Anything that lives in something is a person.
Anything that something lives in is a place.

These patterns require special usage of the words 'anything' and 'something', which accordingly should not be used in names for individuals, classes or properties. The schematic patterns that appear in the options pane are as follows:

Anything that [has-property] something is a [class].
Anything that something [has-property] is a [class].

Step 7

Complex statements using noun-phrase lists

More complex statements about individuals or classes can be composed by linking the phrases already described by three words: the conjunctions 'and' and 'or', and the relative pronoun 'that'. This is done in a controlled way to avoid sentences that are ambiguous.

As an example of the kind of ambiguity we want to avoid, consider this sentence:

> John is a lawyer that lives with a nurse and an artist.

Does John live with two people, or one person who is both nurse and artist? Or is John both a lawyer and an artist?

The first method for constructing a complex sentence is an extension of the patterns described in steps 1 and 2 — simple descriptions of individuals and classes.

London is a city.
A city is an urban area.

Such sentences are formed from an individual or class as subject, followed by the copular verb 'is', followed by an indefinite noun-phrase complement denoting a class. To make the sentence more complicated, you may add 'and' followed by a further indefinite noun-phrase. This can be repeated as many times as you like. The noun-phrases in the list must be linked by 'and', not by commas.

London is a city and a capital and a financial centre.

To obtain options for extending a sentence during editing, delete the full stop at the end of the sentence and insert a space. Schematic patterns for extending the sentence will then appear in the options pane.

London is a city	and a [class]. that [has-property] [Individual]. that [has-property] a [class].

Note that only the *minimal* extensions are listed. However, having chosen the first option and developed the sentence to 'London is a city and a capital', you can again delete the full stop, insert a space, and so obtain the same options for a further extension.

Step 8

Complex statements using verb-phrase lists

The second method for constructing a complex sentence is an extension of the patterns described in steps 3 and 4 — descriptions using object or data properties.

A queen lives in a palace. Queen Elizabeth lives in Buckingham Palace.

To make such sentences more complicated, you can add 'and' followed by another verb-phrase including a property name and an individual/class name (or literal). This can be repeated as many times as desired, always using 'and' rather than a comma.

Queen Elizabeth lives in Buckingham Palace and is aged 85

Note that every item in a verb-phrase list must include a property. The sentence 'Elizabeth lives in a palace and is a queen' is not allowed, since 'is a queen' contains no property. A permitted way of formulating this statement is given in Step 10.

As explained in Step 7, you can obtain options for extending any sentence by deleting the full stop and inserting a space, as in this example:

A queen lives in a palace	and [has-property] [Individual]. and [has-property] [Individual]. that [has-property] [Individual]. that [has-property] a [class].

Choosing either of the first two options in this list will extend the verb-phrase list as just described. (There will also be options including data properties and literals, but for simplicity these are not shown here.)

Step 9

Complex statements using verb-phrase chains

The third method for constructing a complex sentence is an extension of one of the patterns described in step 3 — a description using an object property name followed by a class name.

Queen Elizabeth lives in a palace.

Instead of making a verb-phrase list by continuing with 'and', you can continue with 'that' followed by another verb-phrase including a property name and an individual/class name (or literal). This can be repeated as many times as desired provided that the current sentence ends in a class name (not an individual/literal).

The queen lives in a palace that adjoins a park that is located in London.

Note that having begun a chain, you cannot revert to a list. In other words, having used 'that', you cannot use 'and'; this will lead to ambiguity.

> The queen lives in a palace that is located in London and dates from 1705.

This could mean that the queen rather than the palace dates from 1705. Accordingly, once you are inside a verb-phrase chain, the options for extending the sentence further will all begin with 'that'.

Step 10

Combining lists and chains

You can make even more complicated sentences by combining lists and chains provided you stick to the following rules:

- A noun-phrase list must precede a verb-phrase list or verb-phrase chain.
- A verb-phrase list must precede a verb-phrase chain.

Following these rules will ensure that the sentence is unambiguous.

Example 1: noun-phrase list + verb-phrase list
In Step 8 we noted that the sentence 'Elizabeth lives in a palace and is a queen' is not permitted since 'is a queen' contains no property. The correct way of formulating this statement is as follows:

Elizabeth is a queen that lives in palace.

Further noun-phrases and verb-phrases can now be added while adhering to the rule that noun-phrase lists precede verb-phrase lists.

Elizabeth is a Windsor and a queen that lives in a palace and is aged 85.

Example 2: verb-phrase list + verb-phrase chain
Starting from a verb-phrase list such as 'Elizabeth is aged 85 and lives in a palace', we can attach a verb-phrase chain of any length qualifying the final class name ('palace').

Elizabeth is aged 85 and lives in a palace that is located in London.

Example 3: noun-phrase list + verb-phrase list + verb-phrase chain
Finally, if you wish, you can use all three methods in one sentence, provided that the above-mentioned ordering constraints are met. For example:

> Elizabeth is a Windsor and a queen that is aged 85 and lives in a palace that is located in a city that is named "London".

Step 11

Complex statements using 'or'

Using 'and' (or 'that') and 'or' in the same sentence almost always leads to ambiguity, as in these examples:

> John is a lawyer or an artist and a pet-owner.
> John is a lawyer or an artist that owns a pet.

In one reading, John is definitely a pet-owner, as in 'John is (1) a lawyer or an artist, and (2) a pet-owner. In the other reading he might not be a pet-owner ('John is (1) a lawyer, or (2) an artist and a pet-owner'). As just shown, we can disambiguate these examples with careful punctuation, but this will become cumbersome or break down altogether for more complex sentences.

To avoid such ambiguities by simpler means, OWL Simplified English disallows usage of 'and' (or 'that') and 'or' in the same sentence. At present, just two patterns with 'or' are allowed: noun-phrase lists and verb-phrase lists. Chains are not allowed since 'or' cannot be used with 'that'. Here are some examples of the permitted patterns.

A married person is a husband or a wife.
A student attends a school or attends a college.

Note At present we do not allow a disjunction to serve as the object of a property, as in 'A student attends a school or a college'. This might lead to trouble with numerical quantifiers, as in 'A rowing team contains exactly four men or women', where the OWL interpretation would have to be 'four people, each of whom is either a man or a woman', allowing mixed teams.

Step 12

Advanced editing actions

The simplest way of using the editing tool is to create an ontology by writing a series of sentences, and to save your work either as a text (using *File* | *Save Text*) or in OWL/XML format (using *File* | *Save XML*).

However, if you explore the *File* and *Generate* menus you will find some further actions, allowing you for instance to import an existing OWL/XML file (perhaps produced using a different tool), or to reorganise your text as a glossary with separate sections for each individual, class or property. These actions are described below.

Naming your ontology

You can assign an IRI (a web identifier) to your ontology by the following metadata statement in the text (similar in form to the statement for declaring a verb):

```
# OntologyIRI http://www.ontology.org/ont123
```

Of course the IRI given here is just an example.

Adding comments

Any line starting with '%' will be interpreted as a comment rather than a statement in OWL.

```
% Royal family ontology, started 1st January 2012
```

Importing an ontology

You can load any ontology in OWL/XML format by choosing *File* | *Import XML* and locating the desired XML file. The editor will discard any axioms outside its coverage, and convert the others to Owl Simplified English by deriving individual/class/property names from the relevant OWL identifiers. The colour highlighting will indicate when this conversion has failed to work (e.g., because the rules for forming names have been violated): incorrect sentence parts will be shown in red.

Regenerating a text

The editing tool is able to regenerate all correctly formed sentences starting from their logical form in OWL. The resulting text might differ slightly from yours, because small errors such as 'a animal' rather than 'an animal' will be corrected. To do this choose *Generate* | *Corrected text*. Incorrect sentences, comments, and metadata statements will be left as they are.

Generating a glossary

Ontologies of any significant length are easier to read if statements are organised rather than merely listed in arbitrary order. As an alternative to organising the text by hand, you can regenerate it automatically as a glossary, in which sentences are grouped by topic, and topics are arranged alphabetically. In the re-organised text all metadata statements will be moved to the front, your own comments will be dropped, and new comments will be added to serve as section headings, as in this fragment:

```
% DOG Class(#dog)
A dog is an animal.
A dog lives in a kennel.
No dog is a cat.

% LIVES IN ObjectProperty(#lives_in)
Anything that lives in something is an animal.
Anything that something lives in is a place.
```

To regenerate the text in this form choose *Generate* | *Glossary*.