The questions in this quiz employ a simplified form of OWL, the Web Ontology Language. The aim is to study the ease with which people can understand and use certain aspects of the language. This document describes those features of the language used in the quiz. Here OWL is represented in an English-like format.

1 ELEMENTS OF OWL

For the purposes of the quiz, there are three kinds of entities:

- Classes
- Individuals, i.e. members of classes
- Properties. In this quiz all properties are binary relations between two individuals.

At the beginning of each section of the quiz it will be explained which names refer to what types of entities. The following naming conventions are used throughout the quiz:

- Class names are composed of capital letters. These are sometimes symbolic names, e.g. *A*, and may contain an integer, e.g. *A_1*. They are sometimes meaningful names such as *MALE*.
- Individual names consist of a single lowercase letters, e.g. *a*.
- Properties names consist of lowercase meaningful words. They are either made up of one word, e.g. *implies*, or several words joined together with an underscore, e.g. *has_nearest_neighbour*.

Note that two different names can represent the same entity. There is no assumption that different names represent different entities. This has to be stated explicitly or deduced. This point is returned to in section 3 below.

The next three sections provide more information about classes, individuals, and properties.

2 CLASSES

Classes are introduced with the class statement. For example, the following statement introduces three classes, X, Y and MALE:
Class X, Y, MALE

The order of introduction of the classes here has no significance. The Class keyword simply states that the following names represent classes.

2.1 *and, or & not*

The intersection of two or more classes can be described using the keyword *and*. For example, *A and B and C* means the class of those individuals which are in *A*, in *B* and in *C*.

The union of two or more classes can be described using the keyword *or*. For example *A or B or C* describes the class of those individuals which are either in *A*, or *B* or *C*.

The complement of a class can be described using the keyword *not*. For example, *not A* describes the class of all individuals which are not in *A*.

In general *not* takes precedence over *and*, i.e. *not A and B* requires that *A* is complemented before creating the intersection with *B*. Similarly, *and* takes precedence over *or*, i.e. *A and B or C* requires that the intersection of *A* and *B* is created before forming the union with *C*.

Brackets are sometimes used in expressions with the *and*, *or* and *not* operators, either to enforce a different order from the default order or in some cases simply to aid readability.

2.2 *SubClassOf* & *EquivalentTo*

Classes can have subclasses. For example, we can write *A SubClassOf B*.

Classes can be equated using the *EquivalentTo* keyword, e.g. we could write *A EquivalentTo B*. The *EquivalentTo* keyword can also be used to equate a class name to a class expression, e.g. *A EquivalentTo (B or C or D).*

2.3 *DisjointWith*

DisjointWith is used to indicate that two classes are disjoint, i.e. have no individuals in common. For example, *X DisjointWith Y*, indicates that classes X and Y have no members in common.

2.4 *DisjointUnionOf*

DisjointUnionOf is used to indicate that a class is made up of the union of a number of classes, all of which are disjoint, i.e. have no individuals in common. Thus *A DisjointUnionOf B, C, D* means that *A* is made up of all the individuals in *B*, *C* and *D* and, moreover, that there are no individuals in common between *B* and *C*, *B* and *D*, *C* and *D*.

2.5 *Some* & *Only*

We can use properties, along with the keywords *some* and *only*, to define classes. For example, imagine we have a class *FEMALE*, then *has_child some FEMALE* would mean all people who have some (i.e. one or more) children who are female.

An example of how the *only* keyword is used is: *has_child only FEMALE*. This describes the set of people who, if they have children, only have daughters. Note particularly that this also includes people who have no children at all. Similarly *works only Saturday* would include people who work only on a Saturday, and people who do not work at all.

The *some* and *only* keywords are used in conjunction with *SubClassOf* to indicate that the members of a class possess certain characteristics. For example, we can write *X SubClassOf has_child some PERSON* to indicate that everyone in the class X has one or more children. We could also write *X SubClassOf has_child some FEMALE* to indicate that all members of X have a daughter. Alternatively, we could write *X SubClassOf has_child only FEMALE* to indicate that members of *X* have only a daughter, or no children at all.

We can also use *some* and *only* in conjunction with *EquivalentTo* to define a class, e.g. *HOME_OWNER EquivalentTo owns some HOUSE* defines *HOME_OWNER* to be the class of those who own one or more houses.

Note the difference between the use of *SubClassOf* and *EquivalentTo*. *FATHER SubClassOf has_child some PERSON* allows the possibility that there are individuals, in addition to those in the *FATHER* class, who have one or more children, i.e. mothers. *HOME_OWNER EquivalentTo owns some HOUSE* indicates that the class *HOME_OWNER* is exactly equivalent to the class of owners of one or more houses.

2.6 Unnamed classes

In the examples above with *SubClassOf*, expressions like *has_child some FEMALE*, *has_child only FEMALE* and *has_child some PERSON*, represent unnamed classes, i.e. they define classes which have not been given a name. It is possible to use such unnamed classes in more complex expressions.

For example, we can write:
X *SubClassOf has_child some (has_child some FEMALE)*

This means that all the individuals in X have at least one child who in turn has at least one daughter. Here the expression in brackets, *has_child some FEMALE*, is an unnamed class, which is used to create another unnamed class: *has_child some (has_child some FEMALE)*.

Note that the whole expression is equivalent to the combination of two expressions:
X SubClassOf has_child some Y
Y EquivalentTo (has_child some FEMALE)

or even three expressions:
X SubClassOf Z
Z EquivalentTo has_child some Y
Y EquivalentTo (has_child some FEMALE)

3 INDIVIDUALS

Individuals are introduced with the Individual statement. For example, the following statement introduces the individuals *a* to *e*:
Individual a, b, c, d, e

3.1 *Type*

As explained above, individuals are members of classes. Class membership can be expressed using the *Type* statement. For example, *a Type X* means that the individual *a* is a member of the class *X*.

The *Type* and *not* keywords can be used together to indicate that an individual is not a member of a class; or expressed another way, is a member of the complementary class. For example, *a Type (not X)* states that *a* is not in the class *X*.

3.2 *DifferentFrom*

As already noted, two names can be used for the same individual. If we wish to explicitly state that two names represent different individuals, then we can use the *DifferentFrom* statement, e.g. *a DifferentFrom b*.

3.3 *SameAs*

If, on the other hand, we wish to state that two individuals are the same, then we use the *SameAs* keyword, e.g. *a SameAs b* indicates that the two names *a* and *b* refer to the same individual.

4 PROPERTIES

Properties are introduced with the property statement. For example, the following statement introduces the property has_child:
Property has_child

As already noted, properties are relations between two individuals. The first individual is referred to as the subject, the second is referred to as the object. Thus in the statement *a has_child b*, *a* is the subject and *b* the object.

Properties are introduced with the Property statement. Properties can have characteristics. In this quiz two characteristics are used: transitive and functional.

The characteristic of a property is associated with the property in the Property statement. For example, the following statement introduces the property has_nearest_neighbour and defines it as being functional:
Property has_nearest_neighbour Characteristic Functional

4.1 *Transitive*

A property is transitive if, whenever that property relates *a* to *b* and *b* to *c*, it also relates *a* to *c*. An example of a transitive property is *has_sibling*. If *a has_sibling b* and *b has_sibling c*, then we can conclude that *a has_sibling c*.

4.2 *Functional*

If a property is functional, this means that for any subject there can be only one object. For example, *has_father* could be defined as a functional property, since it is not possible to have more than one father.

Note that using the functional property it may be possible to deduce that two names refer to the same individual. For example, the statements *x has_father y* and *x has_father z* would imply that *y* and z were the same individual.

It may also be possible to deduce that two names refer to different individuals. For example, assume we have the following three statements:
w has_father x
y has_father z
x DifferentFrom z

Then we can deduce that *w DifferentFrom y*. This is because if *w* and *y* referred to the same individual they would have the same father, i.e. *x* refer to the same individual as *z*. However it is stated that *x* and *z* refer to different individuals.

5 FINAL POINTS

Please remember that OWL is a language intended to be interpreted by machines. In the quiz, meaningful names have been chosen for properties and in some cases for classes. However, these are chosen to aid comprehension and memory. The names used do not affect any conclusions to be drawn. All conclusions should be based on the statements in the quiz.

Section 2 noted that brackets are used with *and, or* and *not*, sometimes to alter the order of interpretation of these keywords, sometimes purely to aid readability. Brackets are also used at other places in the OWL statements, in general to aid readability.

In the questions, the OWL keywords, e.g. *Class, SubClassOf, and* etc. are displayed in bold blue to differentiate them clearly from entity names.

6 THE QUIZ

The quiz contains six sections. The first asks for some basic information about you, to aid analysis of the responses. The next four sections contain a total of 34 questions.

Each of the questions contains a number of statements. The final statement is introduced by the $\Rightarrow$ symbol. In each case you are required to decide whether the final statement can be validly concluded from the preceding statements. Please click on the 'Valid' or 'Not valid' button, to indicate your answer. Please note that the order of statements before the $\Rightarrow$ symbol does not affect whether the inference is valid or not; the statements could be presented in any order.

When you are happy with your answer, please click on 'Next' at the bottom of the page. You need to answer each question before moving on to the next question. If you are not sure about an answer, please respond with the answer you think more likely.

There is also a final section which provides an opportunity for you to give feedback about the quiz.

Please do not use your browser 'Back' button at any time. This will cause difficulties in analysing the results. If you feel you have made a mistake in a previous question, please just ignore this and carry on with the quiz.

Many thanks for your help with this study.