JKAU: Comp. IT. Sci., **Vol. 11** No. 1, pp: 13 – 25 (1444 A.H. / 2022 A.D.)
DOI: 10.4197/Comp.11-1.2

CAL2OWL: Direct Translation from CAL to OWL for Ontology Authoring

Hanan Hassan Al Mutawa1 , Hanan Elazhary1,2, Amani Tariq Jamal3 and Nada Bajnaid3

[1]College of Computer Science and Engineering
University of Jeddah, Jeddah, Saudi Arabia
[2]Computers & Systems Department
Electronics Research Institute, Cairo, Egypt
[3]Computer Science Department, Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah, Saudi Arabia
[1,2]{1801276, helazhary}@uj.edu.sa and [3]{atjamal, nbajnaid}@ kau.edu.sa

***Abstract*-- Due to the difficulty of authoring the Web Ontology Language (OWL) by ontology engineers and domain experts with little or no engineering experience, the first Controlled Arabic Language (CAL) was proposed to ease ontology authoring by Arab experts. However, the CAL tool is based on Rabbit to OWL Ontology Language (ROO), meaning that CAL ontologies must be translated to Rabbit before being ultimately translated to OWL, which is a slow and inflexible process. The slowness is due to the intermediate translation step, and the inflexibility is due to the coupling between CAL and Rabbit, which prevents modifying and/or extending CAL statements. This research presents the CAL2OWL ontology authoring tool that has been designed to support CAL by translating ontologies from CAL to OWL directly without passing through Rabbit, making it faster and more flexible. We show how to use CAL2OWL to generate a quite complex Umrah ontology using relatively simple CAL statements and also show equivalent complex OWL statements that would have been written otherwise. A System Usability Scale (SUS) usability test demonstrated that CAL2OWL is also highly usable.**

***Index Terms*—Arabic, CAL, Ontology, OWL, Rabbit, Web Ontology Language**

I. Introduction

The semantic web is an upgraded version of the World Wide Web, which adds semantics to web documents to facilitate processing and comprehension by computers. This means that computers can provide meaningful interpretation in the same way that humans process data to achieve their goals. It involves languages specifically designed to achieve certain goals, e.g., the Resource Description Framework (RDF), the Extensible Markup Language (XML), the Resource Description Framework Schema (RDFS), and the Web Ontology Language (OWL) [1]. The Semantic Web is based on RDF, which is a set of World Wide Web Consortium (W3C) specifications designed as a metadata model. It is used as a general method of conceptual description of web resources in the form of subject–predicate–object triples. XML is a powerful method to format, store, search and share data. The XML language used by RDF is called RDF/XML. RDFS is a semantic extension of RDF. An ontology is a collection of knowledge used to describe a specific domain. OWL is considered the main language of the Semantic Web used for writing ontologies; it provides semantics to the data represented using RDF. Nevertheless, it has more powerful vocabulary and syntax [2].

Because of the complex syntax of such languages, ontology engineers and domain experts need languages to help them in writing and validating ontologies that can be translated to OWL. Therefore, Controlled Natural Languages (CNLs) have been developed. These languages include Controlled Language for Ontology Editing (CLOnE), Attempto Controlled English (ACE), Sydney OWL Syntax (SOS), and Ordnance Survey Rabbit, which is accompanied by the Rabbit to OWL Ontology (ROO) ontology authoring tool [3]. Unfortunately, all these controlled languages serve only the English language. Although identifiers, such as conceptual names and relationships, can be expressed in other natural languages, the keywords are still in English. As a result, the statements will be composed of two languages, making reading, and writing harder and slower. In addition, the error messages are generated in English only, making them hard to understand by users are not proficient in English. This called for developing controlled languages corresponding to various other natural languages.

Specifically, in Arab regions, many domain experts do not have English language proficiency or may find it more convenient to express their knowledge in Arabic. This leads to errors and ambiguities when they attempt to use the existing controlled English languages. Although they can seek help from ontology engineers, they will not be able to verify and validate a resulting ontology because it will be expressed in English. On the other hand, reading or writing a statement with Arabic identifiers and English keywords is difficult and slow. This is because the statements contain words from two different languages, with inherent differences. For example, Arabic is written from right to left, whereas the opposite is true about English. Controlled Arabic Language (CAL) is the first controlled Arabic language introduced to address this problem [4]. Nevertheless, it is based on ROO, meaning that ontologies must be translated to Rabbit before ultimately being translated to OWL, which is a slow process. Modifying and/or extending CAL is not possible under this design.

Another issue is that CAL has not been used to develop a formal world ontology. To address the aforementioned problems, this paper proposes the CAL2OWL tool, which aims at improving the ontology authoring process using CAL. This is achieved by direct translation of CAL statements to OWL to reduce the translation steps employed in CAL tool. This also implies higher flexibility since CAL can be modified and/or extended independently. The paper also proves the efficiencies of CAL and CAL2OWL by using CAL2OWL to develop a complex Umrah ontology using relatively simple CAL statements and showing the equivalent complex OWL statements that would have been written otherwise.

The organization of the rest of the paper is as follows: Section II provides a literature review to prove the importance of CAL2OWL. Sections III and IV explain CAL to OWL mapping and the details of CAL2OWL tool respectively. The developed Umrah ontology is presented in Section V as a case study. CAL2OWL is evaluated in Section VI. Finally, Section VII provides the conclusion and directions for future research.

II. Literature Review

In this section, a review of a set of CNLs proposed in the literature is presented. CNLs have been developed for a limited number of natural languages such as German [5], Spanish [6], and English [7]. In light of the fact that English is one of the most widely used language in the world, the most popular controlled English languages namely, ACE, SOS, CLOnE, and Rabbit, are discussed. This is of course in addition to the controlled Arabic language CAL [4] on which CAL2OWL is based. This is in addition to a recently proposed, but not yet developed, controlled Arabic language.

A. Attempto Controlled English (ACE)

ACE is a controlled English language. It covers a subset of standard English, but constraints the syntax and semantics that can be used. A small set of rules describe the possible constructions and interpretations. It has been under continuous development at the University of Zurich since 1995 [8]. ACE has been developed to allow ontology engineers to express OWL DL (a sublanguage of OWL) in English. It uses the Attempto Parsing Engine (APE) to translate its statements into Discourse Representation Structures (DRSs) that rely on a use a modification of the first-order logic language [9]. ACE allows writing compound statements such as “Japan is a country, and Tokyo is a city” [8]. Unfortunately, using variables such as those in the statement “If X is bigger than Y, then Y is not bigger than X” makes ACE hard for non-experts to understand [10]. There is a first-order reasoner for Attempto Controlled English (ACE). It introduces mathematical and functional extensions. It covers all ACE constructs that have a representation in first-order logic [11].

B. Sydney OWL Syntax (SOS)

Manchester OWL is a formal language with a slightly easier syntax than OWL [12]. SOS is another controlled English language developed based on Manchester OWL. SOS aims at filling the gap between a formal language that is easy for machine processing and a seemingly informal language that is easy for non-specialists to read and write with the help of an intelligent authoring tool [13]. SOS provides bindings to OWL 1 functional syntax. Each OWL functional syntax statement is interpreted as a unit and does not reference any other OWL statement or background knowledge. This facilitates translation from SOS to OWL functional syntax and vice versa. In SOS, a compound statement such as “Japan is a country, and Tokyo is a city” is not allowed and must be represented as two separate statements [13]. However, SOS allows the use of variables such as in the statement “If X is bigger than Y, then Y is not bigger than X”. It also allows nesting of expressions as needed [14]. This makes SOS hard for non-experts like ACE [4].

C. Controlled Language for Ontology Editing (CLOnE)

CLOnE is another controlled English language that has been developed to allow users to design, develop, and manage information in the semantic web without having to learn complicated standards such as XML, RDF, or OWL and without knowledge of typical ontology engineering tools. Though it is a simplified implementation of natural language processors, it permits detailed information representation and allows high accuracy and reliability [15]. The idea behind CLOnE is to accept an input statement regardless of the grammatical agreement. It is designed to either accept a valid input or generate an error message and reject an invalid one [15]. CLOnE has gone through continuous improvement processes and evaluation unlike the standard ontology editor Protégé. In 2021, Preventis & Petrakis presented CLONE as a cloud-based ontology to be used by teams in real-time as a collaborative environment for authoring and editing ontologies. It was designed as a service-oriented architecture in order to benefit from the corresponding easy extensibility and scalability features [16]. Unfortunately, it also allows compound statements because it allows a theoretically unlimited number of concepts or instances per sentence [17]. Hence, it is also hard for non-experts to understand.

D. Rabbit

Rabbit is a fourth controlled English language that has been developed by Ordnance Survey with the goal of assisting domain experts in authoring ontologies [3]. While an ontology is written in English-like statements, OWL DL language (a sublanguage of OWL) features are supported. This allows efficient communication and cooperation between domain experts and ontology engineers. It also allows domain experts to understand a developed ontology and verify it. Additionally, a tool called ROO has been developed as a plugin of Protégé for automatic translation from Rabbit to OWL DL [18]. Rabbit has many features that make it superior to the other English controlled natural languages in terms of ease and simplicity of expressing knowledge with greater details for domain experts with the help of a knowledge engineer. It considers short statements to be less prone to error and more understandable [18]. Accordingly, compound statements such as “America is a country, and Washington is a city” are not allowed in Rabbit. Furthermore, it prohibits the use of variables to maintain its understandability. It also allows an ontology to reference concepts defined in previous Rabbit ontologies [19].

E. Controlled Arabic Language (CAL)

As previously noted, most controlled natural languages are developed in English [20]. They typically allow expressing concepts, their instances, and the relationships in other natural languages such as Arabic, but this does not apply to the keywords. This results in statements formed of two different natural languages which are subsequently slow to read and write [4]. Additionally, error messages are also expressed in English. To address these issues for the Arabic language, CAL has been introduced. In CAL, the whole statement is written in Arabic including the keywords. This also applies to the error messages. CAL is based on the controlled English language Rabbit, and it has a similar syntax, which makes translation between both languages (Rabbit and CAL) easier [4]. Since Rabbit has been developed with domain experts in mind, this is also true about CAL. In other words, CAL is intended to allow Arabic domain experts to easily develop ontologies in Arabic, possibly aided by ontology engineers without having to learn complex languages such as OWL. Unfortunately, CAL tool is based on ROO tool, meaning that ontologies written in CAL must be translated to Rabbit before being ultimately translated to OWL, which is a slow and inflexible process.

F. Arabic Controlled Language

Fahal et al. [21] studied the possibility of developing an Arabic Controlled Language (ACL) using one of two possible approaches. The first is to rely on an already developed language, and the second is to start from scratch. They evaluated both and decided to use the second cleaner approach. Nevertheless, to the best of our knowledge, this has not been achieved yet. It's clear that there is still an urgent need for an efficient tool for authoring ontologies using a controlled Arabic Language. Due to the aforementioned advantages of CAL, we decided to adopt it and improve CAL tool performance through the development of the CAL2OWL tool discussed in this paper.

III. CAL TO OWL MAPPING

To build ontologies, users must declare concepts, instance of concepts, and relationships between them. In this section, CAL and corresponding Rabbit and OWL statements [22] are discussed via a set of examples. It is worth noting that OWL 2 DL is employed by Rabbit and CAL, and hence CAL2OWL. A summary is provided in Table 1.

A. Declaration

Domain concepts should be declared as a first step in developing an ontology.

- Declaring Cupcake (كب كيك in Arabic) as a concept in Rabbit and CAL and as a class in OWL

Example in CAL: كب كيك هو / هي مفهوم
In Rabbit: Cupcake is a concept.
In OWL: Declaration (Class (كب كيك))

- Declaring has topping (له اضافات in Arabic) as a relationship in Rabbit and CAL and as a property in OWL

Example in CAL: له اضافات هو/ هي علاقة
In Rabbit: Has topping is a relationship
In OWL: Declaration (ObjectProperty (له اضافات))

- The keywords *is a* in Rabbit and *هو/هي* in CAL are used to declare an instance of a concept. In OWL, *ClassAssertion* keyword in used.

Example in CAL: فرنسا هو / هي دولة
In Rabbit: France is a country
In OWL: ClassAssertion(فرنسا دولة)

B. Intersection

The keywords *or* in Rabbit and *أو* in CAL are used to express the intersection of objects. In OWL, the subject of the statement will be one item from the set of union of instances of the intersecting concepts.

Example in CAL: كل كب كيك له اضافات شوكولاتة او فراولة او توت
In Rabbit: Every Cupcake has topping Chocolate or Strawberry or Raspberry.
In OWL: SubClassOf (كب كيك ObjectSomeValuesFrom (له اضافات ObjectUnionOf (شوكولاتة فراولة توت)))

C. Union

The keywords *and* in Rabbit and *و* in CAL are used to express the union of objects. In OWL the subject of the statement is an instance of both concepts.

Example in CAL: كل كب كيك توت بالليمون له اضافات توت وليمون
In Rabbit: Every Raspberry Lemon Cupcake has topping Raspberry and Lemon.
In OWL: SubClassOf (كب كيك توت بالليمون ObjectSomeValuesFrom (له اضافات ObjectIntersectionOf (توت ليمون)))

TABLE 1
SUMMARY OF CAL TO OWL MAPPING

OWL 2 Construct/Expression/ Axiom	Rabbit Representation	CAL Representation
Class	Concept	مفهوم
Object/Data Property	Relationship	علاقة
Class/Object Property Assertion	Is a	هو/هي
ObjectIntersectionOf	And	و
ObjectUnionOf	Or	أو
DisjointClasses	No	لا
ObjectSomeValuesFrom	One or more of	واحد او أكثر من
EquivalentClasses	Can only be	يمكن فقط ان يكون/ تكون
EquivalentClasses and ObjectSomeValueFrom	Is anything that:	هو/هي أي شيء الذي/التي
ObjectExactCardinality	Exactly #	بالضبط عدد #
ObjectMinCardinality	At least #	على الاقل عدد#
ObjectMaxCardinality	At most #	على الاكثر عدد #
InverseObjectProperties	Inverse of	عكس
ObjectAllValuesFrom	Only	فقط
ObjectSomeValuesFrom	Every From That	كل من الذي/التي

D. No

The keywords *No* in Rabbit and *لا* in CAL point out that something about a concept and all its instances is not true. OWL represents those classes are disjoint.

Example in CAL: لا كب كيك له اضافات طماطم.
In Rabbit: No Cupcake has topping Tomato.
In OWL: DisjointClasses(كب كيك ObjectSomeValuesFrom (له اضافات طماطم))

E. One or more of

The keywords *one or more of* in Rabbit and *واحد او اكثر من* in CAL are used to represent an object by one or more concepts. In OWL, existential quantification is used to represent existential class expressions via *ObjectSomeValuesFrom*.

Example in CAL: كل كب كيك نوع واحد له اضافات واحد أو أكثر من فراولة، حلويات، كريمة تزيين.
In Rabbit: Every Cupcake Type One has topping one or more of Strawberry, Candy, Frosting Cream.
In OWL: SubClassOf (كب كيك نوع واحد ObjectSomeValuesFrom (له اضافات ObjectUnionOf (فراولة كريمة تزيين حلويات)))

F. Only

The keywords *only* in Rabbit and *فقط* in CAL are used to express that those objects represented in statement are the only possible items that can be used. In OWL, universal quantification is used to represent a universal class expression via the keyword *ObjectAllValuesFrom*.

Example in CAL: كل كب كيك فراولة فقط له اضافات فراولة و كريمة تزيين
In Rabbit: Every Strawberry Cupcake only has topping Strawberry and Frosting Cream.
In OWL: SubClassOf (كب كيك فراولةObjectAllValuesFrom (له اضافات ObjectUnionOf (فراولة كريمةتزيين)))

G. Every

The keywords *Every* in Rabbit and *كل* in CAL are used to indicate that the concepts and their instances have something true. In OWL, existential quantification is used to represent an existential class expression via *ObjectSomeValuesFrom*.

Example in CAL: كل كب كيك له اضافات اضافات كب كيك
In Rabbit: Every Cupcake has topping Cupcake Topping.
In OWL: SubClassOf (كب كيكObjectSomeValuesFrom(له اضافات اضافات كب كيك)))

H. From

The keywords *from* in Rabbit and *من* in CAL are used to specify that the object of the statement is related from whom or from what. In OWL, existential quantification is used to represent an existential class expression via *ObjectSomeValuesFrom*.

Example in CAL: كل كب كيك سعرات منخفضة له وظيفة حماية من السمنة.
In Rabbit: very Low Calories Cupcake has purpose Protection from Obesity.
In OWL: SubClassOf (كب كيك سعرات منخفضة ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf (ObjectSomeValuesFrom(حماية)(السمنة)))

I. is anything that:

The keywords *is anything that:* in Rabbit and *هو/هي أي شيء الذي/التي* in CAL are used to add more details for concepts. OWL uses multi-class expressions to represent this keyword, which are *EquivalentClasses* and *ObjectSomeValueFrom*.

Example in CAL:
الكب كيك شوكولاتة هو/هي أي شيء الذي/التي :
هو/هي نوع من كب كيك,
له اضافات شوكولاتة و كريمة شوكولاتة.
In Rabbit:
A Chocolate Cupcake is anything that:
is a kind of Cupcake;
has topping Chocolate and Chocolate Cream.
In OWL: EquivalentClasses (كب كيك شوكولاته ObjectIntersectionOf (ObjectSomeValueFrom (له اضافات ObjectIntersectionOf ((كريمة الشوكولاتة شوكولاتة) (كب كيك)))

J. Can only be

The keywords *can only be* in Rabbit and *يمكن فقط ان يكون/تكون* in CAL are used to refer to possible specializations of a concept. OWL expresses this via semantically equivalent classes.

Example in CAL: كل كريمة كب كيك يمكن فقط ان يكون/تكون حلاوة زائدة او حلاوة متوسطة او حلاوة خفيفة.
In Rabbit: Every Cupcake Cream can only be Extra Sweetness or Medium Sweetness or Mild Sweetness.
In OWL: EquivalentClasses (كريمة كب كيك ObjectUnionOf (حلاوة خفيفة حلاوة متوسطة حلاوة زائدة)

K. Number constraining keywords

Rabbit and CAL include several keywords for constraining numbers as follows:

- The keywords *exactly #* in Rabbit and *بالضبط عدد #* in CAL are used to constrain specific number of objects that apply. OWL represents those individuals that are connected to an exact number of instances.
- The keywords *at least #* in Rabbit and *على الاقل عدد #* in CAL are used to constrain minimum number of objects that apply. OWL represents those individuals that are connected to a minimum number of instances.
- The keywords *at most #* in Rabbit and *على الاكثر عدد #* in CAL are used to constrain maximum number of objects that apply. OWL represents those individuals that are connected to a maximum number of instances.

An example using *exactly* keyword is shown below. The other number constraining keywords have similar syntaxes.

Example in CAL: كل كب كيك بسيط له اضافات بالضبط عدد 1 اضافات كب كيك.
In Rabbit: Every Plain Cupcake has topping exactly one Cupcake Topping.
In OWL: SubClassOf (كب كيك بسيط ObjectExactCardinality (1 له اضافات اضافات كب كيك)

L. Inverse

The keyword *inverse of* in Rabbit and the keyword *عكس* in CAL are used to express that a relationship is the inverse of another. In OWL, this involves using an inverse object property.

Example in CAL: العلاقة له وظيفة عكس الوظيفة الخاصة ب

In Rabbit: The relationship has purpose is the inverse of is purpose of.
In OWL: InverseObjectProperties (الوظيفة الخاصة ب له وظيفة)

M. That

The keywords *That* in Rabbit and *الذي/التي* in CAL are used to provide the object of the statement a description. In OWL, existential quantification is used to represent an existential class expression via *ObjectSomeValuesFrom*.

Example in CAL: كل كب كيك عيد ميلاد له إضافة شمعة الذي/التي لها وظيفة زينة.
In Rabbit: Every Birthday Cupcake has topping Candle that has purpose Decoration.
In OWL: SubClassOf(كب كيك عيد ميلاد ObjectSomeValuesFrom(له إضافة ObjetIntersectionOf (ObjectSomeValuesFrom(لها وظيفة زينة) شمعة)))

IV. Proposed CAL2OWL Tool Details

The CAL2OWL tool was developed using the Python language. Python was selected because although it is a powerful programming tool, it is easy to use, and maintenance and debugging can be easily handled in it. CAL2OWL tool supports OWL 2 DL with functional-style syntax. Fig.1 depicts the CAL2OWL tool architecture. As shown in the figure, CAL2OWL works in two phases: In the first phase, an input CAL statement is analyzed by the parser. In the presence of an error, a suitable error message expressed in Arabic is output. This process continues until no more errors are detected. In the next phase, the correct CAL statement is translated to OWL. It is an easy-to-use tool, which enables Arabic domain experts to author ontologies without having to master ontology engineering. It can be used with the least amount of training and results in an OWL ontology matching how the domain expert understands the domain. The following samples discuss CAL2OWL interfaces and error messages.

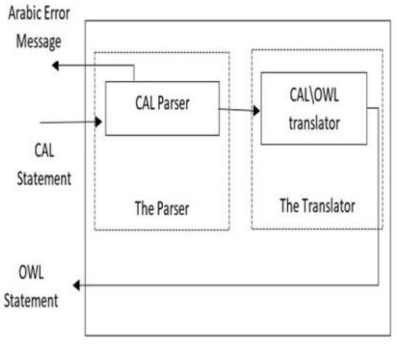


Figure 1 CAL2OWL Tool Architecture

A. CAL2OWL Interfaces

A snapshot of the CAL2OWL interface is depicted in Fig.2. As shown in the figure, the main window contains two frames. The upper frame contains three tabs. The first one (المفاهيم) is used to add concepts to the ontology, the second one (العلاقات) is used to add relationships, and the last one (الجمل) is used to add the statements describing the ontology. The lower frame contains two tabs. The first one (جمل ذات صلة) shows all the ontology statements related to a specific concept in CAL and the other tab (OWL) shows the ontology in OWL. The button حذف قاعدة البيانات at the bottom is used to delete the ontology and restart.

To add a new concept, press the tab المفاهيم then press the button إضافة مفهوم جديد (add a new concept). As shown in Fig.3., two frames are displayed, the first one is for typing the new concept and the other for showing error messages as needed. Adding relationships, and other statements is accomplished via similar steps. Instances of concepts are added form the statements tab (الجمل) based on "هو/هي" keyword. In the main window, the concepts tab (المفاهيم) will show all the concepts that were added to the tool. As shown in Fig.4., next to each concept, there are three buttons: بحث المفهوم (search for concept) for showing all statements related to the concept, *OWL* for showing how the concept is added to the ontology in OWL language, and حذف المفهوم (delete concept) for deleting the concept from the ontology. The same applies to the tabs العلاقات (relationships) and الجمل (statements).

The added concepts, relationships, instances, or statements can be deleted via the حذف (delete) button. The tool converts the input CAL statements into OWL language. To show an OWL statement corresponding to a CAL statement, press the button *OWL*, and to show all the OWL statements, press the *OWL* tab in the lower frame in the main windows.

B. Error Messages

In this section, a sample of the error messages that may be generated to the user in Arabic is presented.

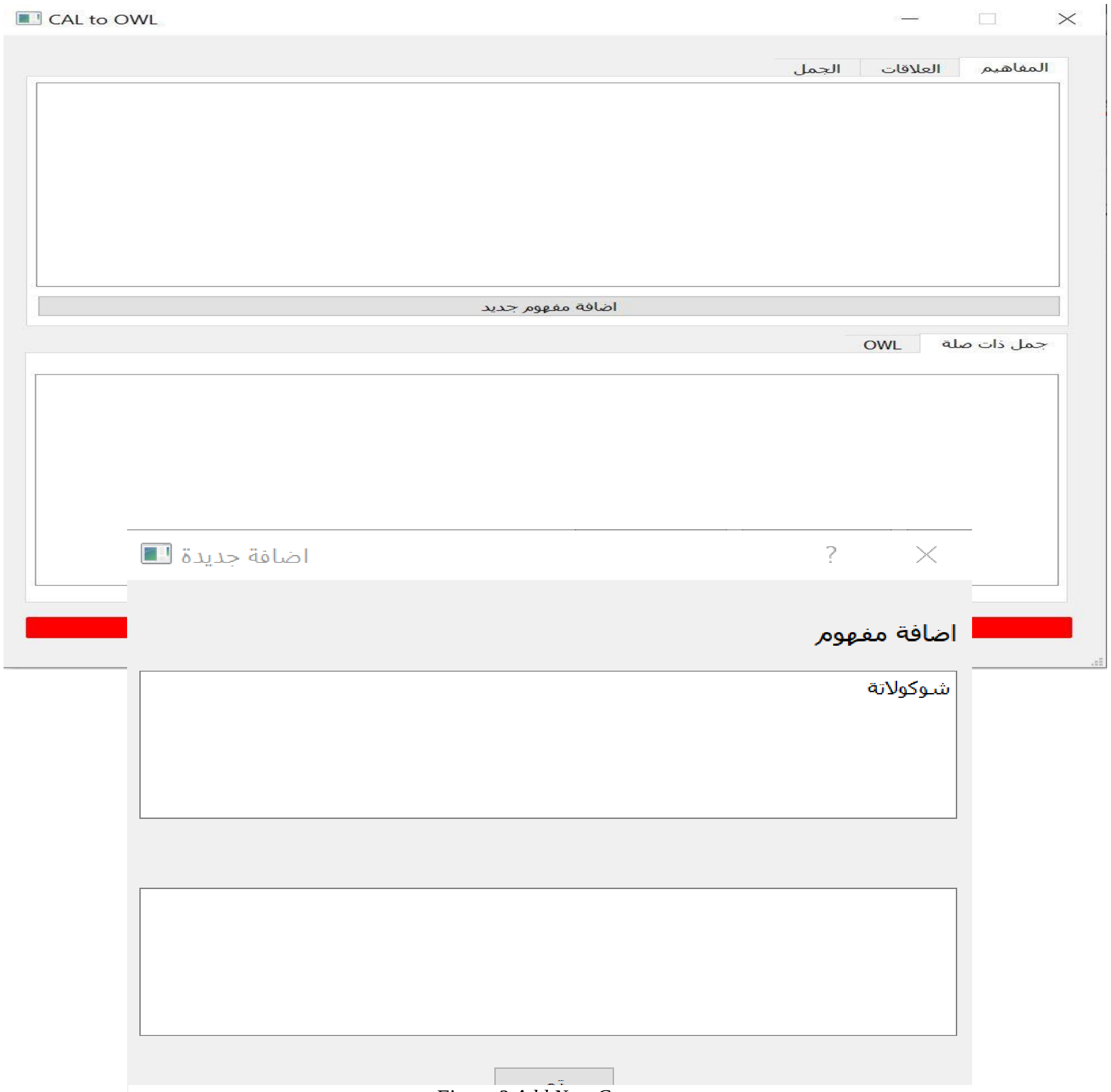
- The message "موجود مسبقا" which means (already exists) will appear when the same concept or relationship is re-entered to the tool. It is depicted in Fig.5.
- The message "المفهوم () غير موجود" (the concept does not exist) will appear when the concept in an input statement is not declared as a concept in the tool. Considering the example "كل كب كيك له إضافات توت و فراولة" (Every Cupcake has topping Strawberry and Raspberry) and assuming it is input without first introducing "توت" (Raspberry) as a concept to the tool, the message "المفهوم (توت) غير موجود" ("the concept does not exist") will appear.
- The message "العلاقة () غير موجودة" (the relationship does not exist) will appear when the relationship in an input statement is not declared as a relationship in the tool. Considering the same example " كل كب كيك له إضافات توت و فراولة" (Every Cupcake has topping Strawberry and Raspberry) and assuming it is input without first introducing "له اضافات" (has topping) as a relationship to the tool, the message "العلاقة (له اضافات) غير موجودة" (the relationship does not exist) will appear.

V. Case Study: Umrah Ontology

In this section, a case study using the CAL2OWL tool to build an Umrah ontology is presented. The goal is to prove that we can develop a quite complex ontology using simple CAL statements and meanwhile, show the complexity of the equivalent OWL statements that would have been used instead.

Fig. 6 depicts the Umrah ontology diagram. The figure shows all the concepts and instances of concepts of the ontology and the relationships between them. The concepts are inserted in circles, the instances are inserted in squares and the relationships are shown using arrows.

Tables 2 and 3 shows a set of CAL statements comprising the Umrah ontology and the corresponding OWL statements. Tables 4, 5 and 6 show the whole Umrah ontology generated by CAL2OWL tool. It is worth noting that it is expressed in OWL2 Functional Style syntax. To generate the ontology, it was first authored using CAL language in a text file, and the file was saved by pressing the button save (حفظ) in the OWL tab in the main window. The time taken to generate the OWL ontology using the CAL2OWL tool is lower than that needed to generate it using the CAL tool. The estimated time reduction was about 45%. This is attributed to the fact that in CAL2OWL the step of translating CAL statements to Rabbit before ultimately translating them to OWL is totally eliminated.



شوكولاتة

Figure 3 Add New Concept

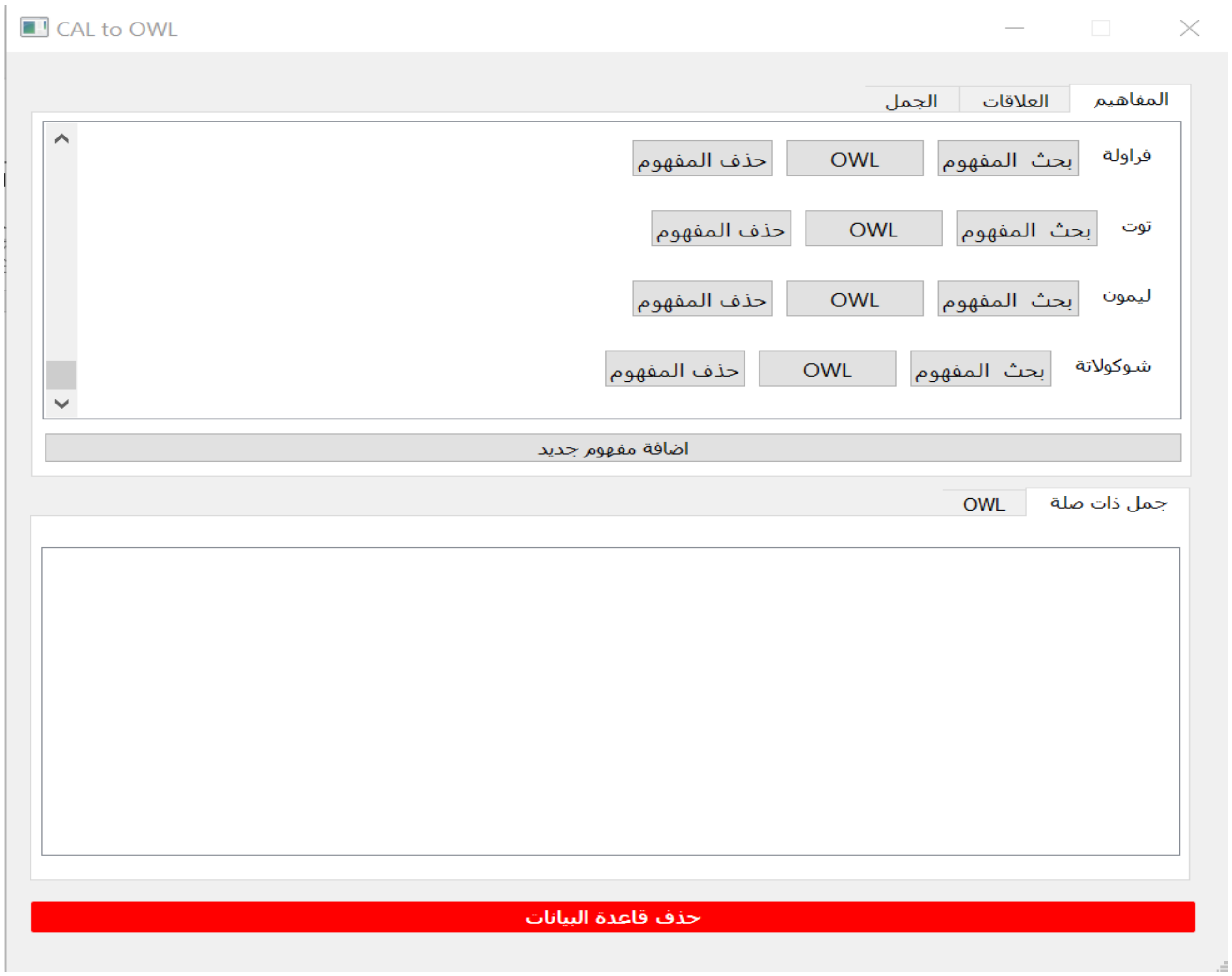


Figure 4 Concepts List

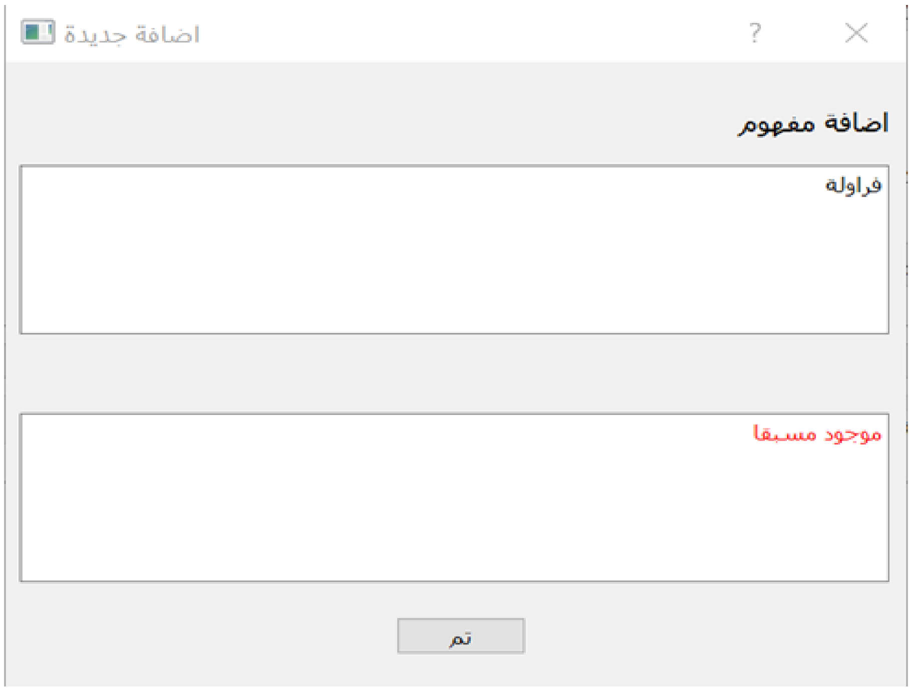


Figure 5 Error Message (already exist)

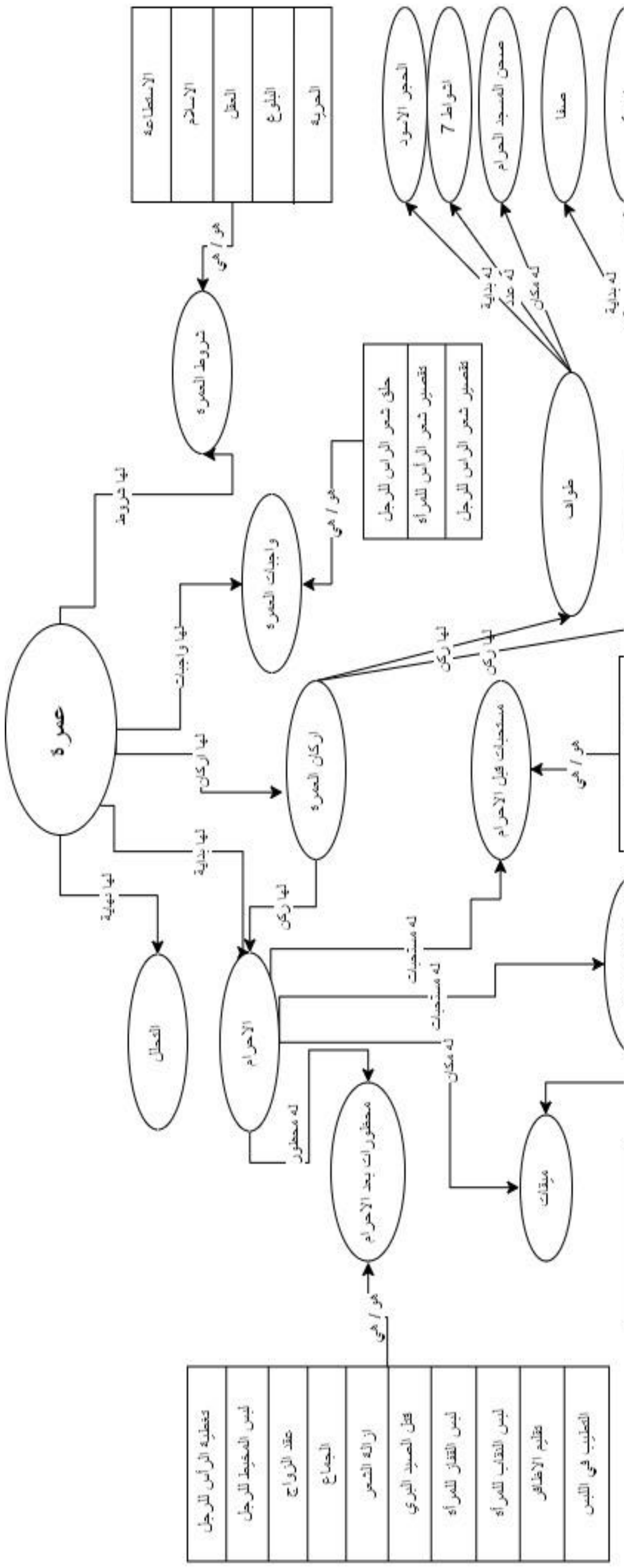


Figure 6 Umrah Ontology Diagram

TABLE 2

UMRAH ONTOLOGY STATEMENTS IN CAL AND OWL (1 OF 2)

CAL statements	OWL statements
كل عمرة لها اركان اركان العمرة	SubClassOf (عمرة ObjectSomeValuesFrom(لها اركان اركان العمرة))
كل عمرة لها شروط شروط العمرة	SubClassOf (عمرة ObjectSomeValuesFrom(لها شروط شروط العمرة))
كل عمرة لها واجبات واجبات العمرة	SubClassOf (عمرة ObjectSomeValuesFrom(لها واجبات واجبات العمرة))
كل عمرة لها واجبات الاحرام من الميقات	SubClassOf (عمرة ObjectSomeValuesFrom (لها واجبات ObjectIntersectionOf (ObjectSomeValuesFrom(الاحرام) (الميقات)))
كل عمرة لها بداية الاحرام من الميقات	SubClassOf (عمرة ObjectSomeValuesFrom(لها بداية ObjectIntersectionOf (ObjectSomeValuesFrom(الاحرام) (الميقات)))
كل عمرة لها نهاية التحلل من الاحرام	SubClassOf (عمرة ObjectSomeValuesFrom(لها نهايةObjectIntersectionOf (ObjectSomeValuesFrom(التحلل) (الاحرام)))
الاحرام له محظورات محظورات بعد الاحرام	SubClassOf (الاحرام ObjectSomeValuesFrom(له محظورات محظورات بعد الاحرام))
الاحرام له مستحبات مستحبات قبل الاحرام و مستحبات بعد الاحرام	SubClassOf (الاحرام ObjectSomeValuesFrom(له مستحباتObjectIntersectionOf (مستحبات قبل الاحرام مستحبات بعد الاحرام)))
الميقات يمكن فقط ان يكون ذا الحليفة او الجحفة او قرن المنازل او يلملم او ذات عرق او مكة	EquivalentClasses (الميقات ObjectUnionOf(ذا الحليفة الجحفة قرن المنازل يلملم ذات عرق مكة))
كل طواف له عدد بالضبط عدد 7 أشواط	SubClassOf (طواف ObjectExactCardinality (7 له عدد أشواط))
كل سعي له عدد بالضبط عدد 7 أشواط	SubClassOf (سعي ObjectExactCardinality (7 له عدد أشواط))
كل طواف له بداية طواف من الحجر الاسود	SubClassOf (طواف ObjectSomeValuesFrom(له بداية ObjectIntersectionOf (ObjectSomeValuesFrom(طواف) (الحجر الاسود)))

TABLE 3
UMRAH ONTOLOGY STATEMENTS IN CAL AND OWL (2 OF 2)

CAL statements	OWL statements
كل طواف له نهاية طواف من الحجر الاسود	SubClassOf (طواف ObjectSomeValuesFrom(له نهايةObjectIntersectionOf (ObjectSomeValuesFrom(طواف) (الحجر الاسود)))
كل سعي له مكان الصفا والمروة	SubClassOf (سعي ObjectSomeValuesFrom(له مكانObjectIntersectionOf (الصفا المروة)))
كل سعي له بداية سعي من الصفا	SubClassOf (سعي ObjectSomeValuesFrom(له بداية ObjectIntersectionOf (ObjectSomeValuesFrom(سعي) (الصفا)))
كل سعي له نهاية سعي من المروة	SubClassOf (سعي ObjectSomeValuesFrom(له نهايةObjectIntersectionOf (ObjectSomeValuesFrom(سعي) (المروة)))
كل طواف له مكان صحن المسجد الحرام	SubClassOf (طواف ObjectSomeValuesFrom(له مكان صحن المسجد الحرام))
ذا الحليفة له وظيفة ميقات من اجل اهل المدينة المنورة	SubClassOf(ذا الحليفة ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf(ObjectSo meValuesFrom(ميقات) اهل المدينة المنورة)))
الجحفة له وظيفة ميقات من اجل اهل الشام و اهل مصر و اهل السودان و اهل المغرب العربي	SubClassOf(الجحفة ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf(ObjectSo meValuesFrom(ميقات) اهل الشام اهل مصر اهل السودان اهل المغرب العربي)))
قرن المنازل له وظيفة ميقات من اجل اهل نجد و دول الخليج العربي	SubClassOf(قرن المنازل ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf(ObjectSo meValuesFrom(ميقات) اهل نجد دول الخليج العربي)))
يلملم له وظيفة ميقات من اجل اهل اليمن واهل جنوب مكة	SubClassOf(يلملم ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf(ObjectSo meValuesFrom(ميقات) اهل اليمن اهل جنوب مكة)))
ذات عرق له وظيفة ميقات من اجل اهل العراق	SubClassOf(ذات عرق ObjectSomeValuesFrom(له وظيفة ObjectIntersectionOf(ObjectSo

	meValuesFrom(ميقات) اهل (العراق)))

TABLE 4
UMRAH ONTOLOGY GENERATED BY CAL2OWL TOOL (1 OF 3)

```
//Class declaration
Declaration(Class(المسعى))
Declaration(Class(صحن المسجد الحرام))
Declaration(Class(المروة))
Declaration(Class(الصفا))
Declaration(Class(اركان العمرة))
Declaration(Class(شروط العمرة))
Declaration(Class(واجبات العمرة))
Declaration(Class(مستحبات قبل الاحرام))
Declaration(Class(محظورات بعد الاحرام))
Declaration(Class(مكة))
Declaration(Class(اهل المدينة))
Declaration(Class( اهل الشام))
Declaration(Class( اهل مكة))
Declaration(Class(اهل السودان))
Declaration(Class(اهل مصر))
Declaration(Class ( اهل العراق))
Declaration(Class(دول الخليج))
Declaration(Class(اهل اليمن))
Declaration(Class(اهل نجد))
Declaration(Class(اهل المغرب العربي))
Declaration(Class(اهل جنوب مكه))
Declaration(Class(الحجر الاسود))
Declaration(Class( أشواط))
Declaration(Class(التحلل))
Declaration(Class(مستحبات بعد الاحرام))
 //Object property:
Declaration(ObjectProperty(له محظور))
Declaration(ObjectProperty(له مستحبات))
Declaration(ObjectProperty(له عدد))
Declaration(ObjectProperty(لها نهاية))
Declaration(ObjectProperty(لها واجبات))
Declaration(ObjectProperty(لها ركن))
Declaration(ObjectProperty(له محظورات))
Declaration(ObjectProperty(له مكان))
Declaration(ObjectProperty(له بداية))
Declaration(ObjectProperty(لها شروط))
Declaration(ObjectProperty(له نهاية))
Declaration(ObjectProperty(له وظيفة))
Declaration(ObjectProperty(لها اركان))
Declaration(ObjectProperty(لها بداية))

Declaration(Class(عمرة))
SubClassOf (عمرةObjectSomeValuesFrom  لها واجبات)
(واجبات العمرة)
SubClassOf ( عمرةObjectSomeValuesFrom (  لها اركان
اركان العمرة ))
SubClassOf (عمرةObjectSomeValuesFrom  (لها نهاية
ObjectIntersectionOf(ObjectSomeValuesFrom(Rabbit:f
rom التحلل )(الاحرام )))
SubClassOf (عمرةObjectSomeValuesFrom(  لها شروط
شروط العمرة))
SubClassOf (عمرةObjectSomeValuesFrom (  لها واجبات
ObjectIntersectionOf(ObjectSomeValuesFrom(Rabbit:f
rom الاحرام) (ميقات )))
```

TABLE 5
UMRAH ONTOLOGY GENERATED BY CAL2OWL TOOL (2 OF 3)

```
Declaration(Class( ميقات))
EquivalentClasses  (ميقات ObjectUnionOf( الجحفة
ذا الحليفة
ذات عرق
قرن المنازل
مكة
يلملم))
Declaration(Class(سعي))
SubClassOf (سعي ObjectSomeValuesFrom  له مكان)
(المسعى)
SubClassOf (سعي ObjectExactCardinality(7   له عدد
أشواط))
SubClassOf ( سعي ObjectSomeValuesFrom ( له نهاية
ObjectIntersectionOf(
ObjectSomeValuesFrom(Rabbit:from سعي)(المروة )))
SubClassOf سعي)ObjectSomeValuesFrom (  له بداية
ObjectIntersectionOf(
ObjectSomeValuesFrom(Rabbit:from سعي )(الصفا)))

Declaration(Class(طواف))
SubClass  ( طواف ObjectExactCardinality( 7   له عدد
أشواط))
SubClassOf ( طواف ObjectSomeValuesFrom(  له
نهايةObjectIntersectionOf (
ObjectSomeValuesFrom(Rabbit:fromطواف)( الحجر
الاسود)))
SubClassOf ( طواف ObjectSomeValuesFrom(  له
بدايةObjectIntersectionOf (
ObjectSomeValuesFrom(Rabbit:fromطواف)( الحجر
الاسود)))
SubClassOf (طواف ObjectSomeValuesFrom  له مكان)
(صحن المسجد الحرام)

Declaration(Class(الاحرام))
SubClassOf (الاحرام ObjectSomeValuesFrom له)
(محظورات  محظورات بعد الاحرام)
SubClassOf ( الاحرام ObjectSomeValuesFrom له مستحبات
(ObjectIntersectionOf (  مستحبات قبل الاحرام    مستحبات بعد
الاحرام)))

ClassAssertion(يلملم ميقات)
SubClassOf(  يلملم ObjectSomeValuesFrom ( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom( ميقات)(
اهل العراق)))
ClassAssertion(مكه  ميقات)
SubClassOf (  مكه ObjectSomeValuesFrom( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom( ميقات
)(اهل مكه)))
ClassAssertion(ذا الحليفة ميقات)
SubClassOf (  ذا الحليفة ObjectSomeValuesFrom( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom( ميقات
)(اهل المدينة)))
```

```
ClassAssertion(ذات عرق ميقات)
SubClassOf( ذات عرق ObjectSomeValuesFrom( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom( ميقات)
(اهل العراق)))
```

TABLE 6

Umrah Ontology Generated by CAL2OWL Tool (3 of 3)

```
ClassAssertion(الجحفة ميقات)
SubClassOf( الجحفة ObjectSomeValuesFrom( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom ميقات) اهل
الشام اهل مصر اهل السودان ) اهل المغرب العربي)))
 ClassAssertion(قرن المنازل ميقات)
SubClassOf( قرن المنازل ObjectSomeValuesFrom( له وظيفة
ObjectIntersectionOf(ObjectSomeValuesFrom(ميقات)(
دول الخليج العربي اهل نجد)))

ClassAssertion(الاستطاعة شروط العمرة)
ClassAssertion(الحرية شروط العمرة)
ClassAssertion(الاسلام شروط العمرة)
ClassAssertion(العقل شروط العمرة)
ClassAssertion(البلوغ شروط العمرة)
ClassAssertion(التطيب في البدن مستحبات قبل الاحرام)
ClassAssertion(تقليم الاظافر مستحبات قبل الاحرام)
ClassAssertion(حلق شعر العانه مستحبات قبل الاحرام)
ClassAssertion(الاغتسال مستحبات قبل الاحرام)
ClassAssertion(التلبية مستحبات بعد الاحرام)
ClassAssertion(تقليم الاظافر محظورات بعد الاحرام)
ClassAssertion(التطيب في اللبس محظورات بعد الاحرام)
ClassAssertion(لبس المخيط للرجل محظورات بعد الاحرام)
ClassAssertion(إزالة الشعر محظورات بعد الاحرام)
ClassAssertion(الجماع محظورات بعد الاحرام)
ClassAssertion(عقد الزواج محظورات بعد الاحرام)
ClassAssertion(قتل الصيد البري محظورات بعد الاحرام)
ClassAssertion(لبس القفاز للمرأة محظورات بعد الاحرام)
ClassAssertion(تغطية الرأس للرجل محظورات بعد الاحرام)
ClassAssertion(لبس النقاب للمرأة محظورات بعد الاحرام)
ClassAssertion(تقصير شعر الرأس للرجل واجبات العمرة)
ClassAssertion(حلق شعر الرأس للرجل واجبات العمرة)
ClassAssertion(تقصير شعر الرأس للمرأة واجبات العمرة)
```

VI. EVALUATION & DISCUSSION

To evaluate the proposed tool, the System Usability Scale (SUS) Questionnaire [23, 24] is employed. This questionnaire has proven itself in several usability tests over the years. The questionnaire is used because the questions were appropriate for evaluating the tool. Table 7 shows the items on the questionnaire. On a Likert scale [25], the participants gave answers to the questions ranging from one (strongly disagree) to five (strongly agree). The table provides the mode of each question response, which is 1 for the odd questions (ideal) and 5 for the even ones (ideal) indicating very high usability. Assuming the scores are equidistant, the table also provides the mean of each question response.

Computing the SUS score per respondent involves reducing the score of each odd questionnaire item by 1 to obtain a number in the interval [0, 4]. Adding the five adjusted scores results in the odd SUS score, SUS_o as shown in the following equation:

$$SUS_o = \sum_{i=1,i\ odd}^{9}(s_i - 1) \quad (1)$$

This is followed by subtracting the score of each even questionnaire item from 5. Adding the five adjusted scores results in the even SUS score, SUS_e as follows:

$$SUS_e = \sum_{j=2,j\ even}^{10}(5 - s_j) \quad (2)$$

Finally, the respondent SUS score, ranging from 0 to 100, is computed as follows:

$$SUS = (SUS_o + SUS_e) * 2.5 \quad (3)$$

The computed scores ranged from 80 to 100 with an average of 90.25 indicating that the tool is highly usable.

TABLE 7
RESULTS OF THE USABILITY QUESTIONNAIRE

#	Indicator of evaluation	Mode	Mean
1	I believe that I would like to employ this tool regularly	5	4.4
2	I found the tool pointlessly complicated	1	1.1
3	I believe the tool is pretty easy to use	5	4.8
4	I believe I will seek technical support to be able to use this tool	1	1.3
5	I believe the different functions in this tool are really well integrated	5	4.7
6	I believe there are many inconsistencies in the tool	1	1.5
7	I believe that most users can learn how to use the tool pretty fast	5	4.7
8	I found using the tool very unwieldy	1	1.3
9	I felt very self-confident using the tool	5	4.8
10	I had to learn many things before using the tool I could start using this tool	1	2.1

The results of the usability test were expected due to the obvious simplicity of CAL statements in comparison to OWL. In addition to the usability test, the generated Umrah ontology has been validated by a domain expert. Nevertheless, the main benefit of CAL2OWL is the speedup gained due to eliminating the step of translating CAL statements to Rabbit before ultimately translating them to OWL. This is in addition to the flexibility gained by decoupling CAL from Rabbit allowing modifying and/or extending CAL independently as needed.

VII. CONCLUSION AND FUTURE WORK

This paper presented the CAL2OWL tool for authoring Arabic ontologies by direct translation from the controlled Arabic language CAL to OWL. It serves Arabic domain experts who prefer to author their ontologies in Arabic. Error messages are also generated in Arabic to facilitate the experts' understanding. CAL2OWL is an improvement of the CAL tool in which CAL statements were translation first to the controlled English language Rabbit and then to OWL. Due to eliminating this step, time needed to generate a given ontology is greatly reduced. Additionally, this decoupling of CAL and Rabbit allows us to add/modify CAL statements independently. To the best of our knowledge, this is the first and, at the time of this study, the only tool developed for this purpose and with these capabilities.

A case study was presented to show how we could efficiently use simple CAL statements to develop a quite complex Umrah ontology, and to show the complexity of the corresponding OWL statements that would have been written instead. We also conducted a SUS usability test, which depicted that the tool is highly usable. This is in addition to validating the resulting ontology with the help of a domain expert.

As future work, we intend to study the possibility of expanding CAL2OWL with additional statements independent of Rabbit, given that they have been now decoupled. The goal is to provide a fully functioning tool to promote the development of ontologies in Arabic as needed. Developing more complex ontologies will be also attempted. This is in addition to formal verification and validation of the tool.

References

[1] Harth, A., Janik, M. & Staab, S. (2011). Semantic web architecture. In *Handbook of Semantic Web Technologies.*

[2] Antoniou, G. & Van Harmelen, F. (2004). Web ontology language: OWL. In *Handbook on Ontologies* (pp. 67-92). Springer, Berlin, Heidelberg.

[3] Hart, G., Johnson, M., & Dolbear, C. (2008). Rabbit: Developing a control natural language for authoring ontologies. In *European Semantic Web Conference* (pp. 348-360). Springer, Berlin, Heidelberg.

[4] Elazhary, H. (2016). CAL: A controlled Arabic language for authoring ontologies. *Arabian Journal for Science and Engineering*, 41(8), 2911–2926.

[5] Schwitter, R. (2005). A controlled natural language layer for the semantic web. In *Australasian Joint Conference on Artificial Intelligence* (pp. 425-434). Springer, Berlin, Heidelberg.

[6] Calderón, S. (2015). Building a Controlled Natural Language Framework for Real-time Machine Translation. *Revista de Lenguas Modernas*, (23).

[7] Fuchs, N., Kaljurand, K. & Kuhn, T. (2008). Attempto controlled English for knowledge representation. *In Reasoning Web* (pp. 104-124). Springer, Berlin, Heidelberg.

[8] Schwitter, R. (2010). Controlled natural languages for knowledge representation. In *Coling 2010: Posters* (pp. 1113-1121).

[9] Fuchs, N. & Kaljurand, K. (2006). Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces.

[10] Fuchs, N. E., Kaljurand, K., & Kuhn, T. (2008). Attempto Controlled English for knowledge representation. In *Reasoning web* (pp. 104-124). Springer, Berlin, Heidelberg.

[11] Fuchs, N. (2021). Reasoning in Attempto Controlled English: Mathematical and functional extensions. In *Proceedings of the Seventh International Workshop on Controlled Natural Language* (CNL 2020/21).

[12] Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C. & Hart, G. (2008). A comparison of three controlled natural languages for OWL 1.1.

[13] Cregan, A., Schwitter, R., & Meyer, T. (2007). Sydney OWL Syntax-towards a Controlled Natural Language Syntax for OWL 1.1. In *OWLED* (Vol. 258).

[14] Davis, B. P. (2013). *On applying controlled natural languages for ontology authoring and semantic annotation* (Doctoral dissertation).

[15] Tablan, V., Polajnar, T., Cunningham, H., & Bontcheva, K. (2006). User-friendly ontology authoring using a controlled language. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

[16] Preventis, A., & Petrakis, E. G. (2021). CLONE: Collaborative Ontology Editor as a Service in the Cloud. *Procedia Computer Science*, *184*, 275-282.

[17] Davis, B., Iqbal, A. A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., & Handschuh, S. (2008). Roundtrip ontology authoring. In *International Semantic Web Conference* (pp. 50-65). Springer, Berlin, Heidelberg.

[18] Engelbrecht, P., Hart, G., & Dolbear, C. (2009). Talking rabbit: a user evaluation of sentence production. In *International Workshop on Controlled Natural Language* (pp. 56-64). Springer, Berlin, Heidelberg.

[19] Denaux, R., Holt, I., Dimitrova, V., Dolbear, C., & Cohn, A. G. (2008). Supporting the construction of conceptual ontologies with the ROO tool. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC), Washington* (pp. 1-2).

[20] Kuhn, T. (2014). A survey and classification of controlled natural languages. *Computational Linguistics*, 40(1), 121-170.

[21] El Fahal, H. S., Nasri, M., Bouzoubaa, K., & Kabbaj, A. (2019). Roadmap for an Arabic Controlled Language. *International Journal of Information Technology and Language Studies*, *3*(3).

[22] McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation*, *10*(10), 2004.

[23] Brooke, J. (1996). SUS: A quick and dirty Usability Scale. *Usability Evaluation in Industry*, 189(3).J.

[24] Brooke, J. (2013). SUS: A retrospective. *Journal of Usability Studies*, 8(2), 29-40.

[25] Dharmarajan, B., & Gangadharan, K. (2013). Applying technology acceptance (TAM) model to determine the acceptance of nursing information system (NIS) for computer generated nursing care plan among nurses. *Int J Comput Trends Technol*, *4*(8), 2625-2629.

CAL2OWL: اداة للتحويل المباشر من لغة CAL الى OWL لكتابة الانطولوجيا

حنان حسن آل مطوع[1]، حنان الأزهري[1,2]، اماني جمال[3]، ندى بجنيد[3]
كلية علوم الحاسب والهندسة [1]
جامعة جدة، جدة، المملكة العربية السعودية
أجهزة الحاسوب والأنظمة
معهد بحوث الإلكترونيات، القاهرة، مصر
قسم علوم الحاسب، كلية الحاسبات وتقنية المعلومات [3]
جامعة الملك عبد العزيز، جدة، المملكة العربية السعودية

المستخلص. نظرًا لصعوبة كتابة الأنطولوجيا بلغة الويب (OWL) من قبل مهندسي الأنطولوجيا وخبراء المجال الذين لديهم خبرة هندسية قليلة أو معدومة، تم اقتراح أول لغة عربية لكتابة الأنطولوجيا (CAL) لتسهيل الكتابة من قبل الخبراء العرب. أداة (CAL) تستند إلى لغة Rabbit to OWL Ontology (ROO) ، مما يعني أنه يجب ترجمة الأنطولوجيا من لغة (CAL) إلى لغة (Rabbit) قبل ترجمتها إلى (OWL) ، وهي عملية بطيئة وغير مرنة. بسبب خطوة التحويل الزائدة التي تؤدي إلى عدم المرونة بسبب الاقتران بين لغتي(CAL) و (Rabbit)، مما يمنع تعديل و / أو توسيع عبارات (CAL) يقدم هذا البحث أداة كتابة الأنطولوجيا (CAL2OWL) التي تم تصميمها لدعم (CAL) من خلال ترجمة الأنطولوجيا من (CAL) إلى (OWL) مباشرة دون المرور عبر (Rabbit) ، مما يجعلها أسرع وأكثر مرونة. نعرض في هذه الورقة كيفية استخدام اداة (CAL2OWL) لبناء أنطولوجيا عمرة معقدة للغاية باستخدام عبارات (CAL) بسيطة نسبيًا ونعرض أيضًا في هذه الورقة عبارات (OWL) المعقدة المكافئة التي كان من الممكن كتابتها بطريقة أخرى. وقد أظهر اختبار مقياس قابلية استخدام النظام (SUS)الذي تم تطبيقه على الأداة بأنها قابلة للاستخدام بشكل كبير.