Controlled and Uncontrolled English for Ontology Editing

Brian Donohue
CUBRC
Buffalo, NY

Robert Ganger
CUBRC
Buffalo, NY

Tien Pham
Army Research Lab
Adelphi, MD

Amardeep Bhattal
IBM
Southampton, UK

Barry Smith
University at Buffalo
Buffalo, NY

Douglas Kutach
CUBRC
Buffalo, NY

Ron Rudnicki
CUBRC
Buffalo, NY

Geeth de Mel
IBM
White Plains, NY

Dave Braines
IBM
Portsmouth, UK

***Abstract*—Ontologies formally represent reality in a way that limits ambiguity and facilitates automated reasoning and data fusion, but is often daunting to the non-technical user. Thus, many researchers have endeavored to hide the formal syntax and semantics of ontologies behind the constructs of Controlled Natural Languages (CNLs), which retain the formal properties of ontologies while simultaneously presenting that information in a comprehensible natural language format. In this paper, we build upon previous work in this field by evaluating prospects of implementing International Technology Alliance Controlled English (ITA-CE) as a middleware for ontology editing. We also discuss at length a prototype of a natural language conversational interface application designed to facilitate ontology editing *via* the formulation of CNL constructs.**

Keywords—Ontology; Controlled English; Intelligence Collection

I. INTRODUCTION

Ontologies formally represent reality in a way that limits ambiguity and facilitates automated reasoning and data fusion. Many technologies are available for building, sharing, and using ontologies, including Web Ontology Language (OWL) and controlled natural languages (CNLs). On the one hand, OWL provides effective representation constructs and enables efficient reasoning procedures but is daunting to the non-technical user. On the other hand, CNLs, which are restricted versions of natural languages, provide a human-friendly representation format that is easier for non-technical users but there is no established standard for how statements of CNLs should map onto assertions defining an ontology.

Motivated by the accessibility of CNL, we explored how to create software infrastructure that would enable users to interact with an OWL knowledge base through CNL constructs. We see the value of such an infrastructure for the intelligence, defense and security communities as being realized in the use of ontology-driven information collection applications. Such applications typically have one of two opposing shortcomings. First, users can be prevented from entering information about an entity if that type of entity is not represented in the underlying ontology, but this can be perceived by users as unfriendly. Second, applications can let users enter such information, but this allows the ontology to be modified in ways that do not follow best practices.

In this report, we specifically examine the interplay between OWL 2 DL [1] and the International Technology Alliance's Controlled English, ITA-CE [2]-[6] to determine the feasibility of using ITA-CE as a medium through which ontologies can be correctly modified by non-technical users.

There are two main conclusions we have drawn from this investigation. First, a conversational interface application can assist users in ontology editing tasks. We developed a prototype software application that can be used either as a command-line application or as part of the conversational panel found in IBM's ITA-CE processing environment called 'CE Store'. Our application allows people to converse with a computer in everyday English so that the user's intentions regarding ontology can be rendered into a CNL equivalent. The CNL command can then be passed on to additional machine agents, which modify the ontology and store the result in an OWL file. The application can easily be extended to ingest other data formats such as relational databases and ontology formats other than OWL. At the same time, we discovered that such software does not strictly require the use of CE statements in general or of ITA-CE statements in particular. However, within the current implementation of the conversational interface application, ITA-CE is presented to human users as unambiguous confirmation prompts to ensure that the user's natural language commands were interpreted correctly.

The second conclusion was that ITA-CE might serve as a convenient communication medium for analysts and developers handling information sources in a variety of formats, and by enabling machine agents to exploit additional information sources when attempting to interpret requests made by users.

The rest of the document is structured as follows. Section II discusses existing approaches to utilizing CNL-based approaches to modifying ontologies. Section III provides an

overview of ITA-CE. In Section IV, we provide a detailed discussion of our implementation to support conversational OWL ontology editing. In Section V, we introduce a few illustrative scenarios to show the applicability of our work. We conclude in Section VI by sketching future directions of our work.

II. Controlled Natural Language Ontology Editors

The formal underpinnings of semantic technologies are substantial obstacles for a casual end-user. This usability problem has been widely noted already within the Semantic Web. For example, Rector [7] documented numerous errors commonly made by non-expert ontology users. These include (1) the failure to make all information explicit, (2) ignorance of the effects of range and domain restrictions, (3) mixing up defined and primitive classes, (4) misunderstanding common logical constructs ('and', 'or', 'some not', 'not some'), (5) presuming that classes are disjoint by default, and (6) being insensitive to open world reasoning. Thus, non-expert ontology users face an immense hurdle developing and utilizing ontology-based information sources.

In response to the usability problem, previous research [8]-[14] has sought to hide the formal syntax and semantics of ontologies behind CNL constructs. Several projects in particular have sought to exploit existing CNLs, or develop new CNLs, in order to simplify the tasks of creating, managing, and navigating ontologies.[1] Several software applications now allow users to edit ontologies by writing English sentences that are restricted in admissible vocabulary and grammatical constructions, yet relatively easily comprehended. This sets them apart from traditional tree-structured and graph-structured ontology editors such as Protégé and TopBraid Composer™.

To a great extent, these CNL ontology editors have helped to bridge the gap between casual users and ontological formalisms. However, their success is limited by at least one of four recalcitrant problems. First, most of the editors are not fully compliant with OWL 2 DL, the most widely used member of the OWL family of languages. ACE View does not currently support sentences that express data properties and their corresponding datatypes (e.g., having a date-time value associated with an event). GINO Editor and CLOnE Editor restrict users to only very basic OWL constructs (e.g., there appears to be no support for inserting class axioms or cardinality restrictions). The ROO Editor employs a more expressive language than ACE View, GINO Editor, and CLOnE Editor, but it shies away from OWL 2 DL expressivity for simplicity [12]. Only the Fluent Editor™ 2014 connects users with OWL2 DL's full expressive potential.

Second, some of the editors employ insufficient resources for explaining and correcting user input error. CLOnE Editor users, for example, complained of receiving little guidance for inputting CNL expressions and no feedback to explain syntactical errors [11]. Similarly, users of ROO Editor complained of receiving no feedback on semantic errors [13].

Third, the documented experiments conducted with CNL ontology editors suggest that all but the simplest editing tasks lie beyond the ordinary capabilities of the CNL ontology editor user. Of the editors surveyed above, GINO, CLOnE, and ROO ran experiments with non-expert ontology users. In each case, however, users were successful only in performing basic tasks (e.g., most users could create a class, property, or instance), and for the most part unsuccessful in executing any more sophisticated task (e.g., correctly adding an axiom). If the usability problem is to be solved altogether, then end-users will need an interface that relieves much of the burden of expressing the wide array of OWL constructs.

Fourth, all of the editors require end-users to master the stringent lexical and syntactic rules governing the implemented CNL. For instance, users must be sensitive to quantifiers ('every', 'some'), disallowed terms ('or', 'not'), and peculiar lexical conventions (e.g., in ACE, dashes between multiple elements of a term, e.g., 'stretch-of-river'). Although the GINO and Fluent editors assist users in entering CNL through predictive mechanisms, and although CLOnE and ROO employ less stringent lexical and syntactic restrictions, the fact remains that users are expected to write impeccable CNL sentences. Thus, even if CNL-based editors are more human friendly than traditional tree-structured or graph-structured ontology editors, they nevertheless run the risk of alienating the non-expert user, as put forward by Smart [15].

Related projects include formulations of alternative OWL syntaxes, aimed at simplifying OWL for non-expert users, and resulting in an ontology language that resembles, in many of its properties, a CNL. In particular, we note Manchester OWL Syntax [16], which is recognized by W3C, and Sydney OWL Syntax [17].

The software discussed in this paper aims to overcome the usability problem in a different way. If it were fully developed, it would minimize people's direct interaction with CNLs and eliminate altogether the requirement for them to write syntactically correct sentences of a CNL. Instead, a non-technical user could engage in ontology creation, editing, and management entirely by means of a natural language human-machine conversation. At most a user would need to read a sentence of CNL in order to confirm that the conversational agent has correctly interpreted the natural language input. Behind the scenes, the conversational agent would translate the user's

[1] There has been little comparative study of the vast array of CNLs actively in use. Kuhn (2014) surveys 100 recent CNLs, and Schwitter (et al., 2008) offers a detailed comparison of three of the more prominent CNLs in use today: Attempto Controlled English (ACE), Ordnance Survey Rabbit, and Sydney OWL Syntax (SOS).

commands into a CNL, which would then be passed along to other agents that have access to particular knowledge representation documents, e.g., an OWL/XML file. In our current preliminary implementation of this idea, all of the interaction can take place within the CNL processing environment designed by the International Technology Alliance (ITA) and within the context of previous ITA research on CNL-based tools and their military applications.

III. Overview of ITA Controlled English

In 2010, the International Technology Alliance (ITA) began developing a CNL known as ITA Controlled English (ITA-CE) for the purpose of supporting tasks within the Data-to-Decisions (D2D) framework, "specifically to assist coalition decision makers in distributed information environments through automated or semi-automated fusion processes" [4]. Previous ITA-CE research has addressed the problem of the miscommunication between US and UK military personnel rooted in lexical and cultural discrepancies and the problem of enhancing shared understanding and communication for military decision-making, especially through the exploitation of sensor resources. An example of this is MOIRA (Mobile Intelligence Reporting App), which aims to expedite data requisition within ISR missions.

At base, the syntax and semantics of ITA-CE are adopted from Sowa's Common Logic Controlled English [18]-[19], which in turn aligns itself closely to first-order logic. As with most CNLs, the resultant expressions of the language are readily comprehensible to ordinary English speakers. For example:

```
there is a person named Steve.
the person Steve is married to the person Jane.
the person Steve has the person Jane as spouse.
```

By writing ITA-CE sentences, users can gradually construct a model, in which all pertinent entities within that model – including types, properties, relations, and individuals – are specified. Thus, to construct a model, a user would write sentences defining the objects, properties, and relationships within that domain.

New terms are introduced to the model by means of "conceptualise" statements. For example:

```
conceptualise a ~ Chihuahua ~ C that is a dog.
conceptualise the Chihuahua C ~ barks at ~ the
person P1.
conceptualise the Chihuahua C has the person P2 as
~ owner ~.
```

The term being added or modified is set off by tildes (~), and if the new term is a noun, it is followed by an uppercase variable name.

Additionally, users can write rules of inference to the model in the form of "if-then" statements. For example:

```
if
  ( the person P1 is married to the person P2 )
then
  ( the person P2 is married to the person P1 ) .
```

This rule states that the "is married to" relation is symmetric. If a user includes this rule in a model and also includes the sentence "the person Steve is married to the person Jane," then software can easily and correctly infer "the person Jane is married to the person Steve."

Models can be defined and extended using the ITA-CE processing environment, called 'CE Store' [5].[2] ITA-CE's CE Store software allows users to define and execute custom "CE agents," including conversational agents with which users can converse in ordinary English. Below, we present a pair of CE agents, one of which assists the user in formalizing natural language expressions and the other of which retrieves and emends information within OWL files.

IV. Conversational Interface Application

The conversational interface application that we coded cooperates with the CE Store software, which takes the form of a web application that accepts ITA-CE sentences as input and responds by remembering the ITA-CE sentence and possibly triggering other behavior based on the content of the ITA-CE sentence. Our software is an add-on to the CE Store in the sense that it is a WAR file that can be placed in the same folder as the CE Store's WAR files and is configured by storing a few CE statements in the CE Store.

Given the suitability of the CE Store for incorporating ontology information, one possible strategy for editing ontologies by means of ITA-CE is first to translate the English into ITA-CE and then ITA-CE into OWL. In practice, however, such a strategy faces some obstacles.

To see why, first observe that in order for the software to guide the user in editing an ontology, it must somehow access all the relevant information about the ontology. Because the CE Store software stores its data in the form of ITA-CE statements, the ontology would first need to be loaded into CE Store. This would incur some data redundancy and would require the addition of some rules to draw inferences not explicitly asserted by the ontology. Neither of these is a sizable obstacle. However, if we were to adopt this strategy, and then a user made changes to the ontology, then the previous statements might become false. For example, an ontology might define a hierarchy in which Z is a direct subtype of X. If a user chooses to insert a new type Y so that X is a parent of Y and Y is a parent of Z, the previous statement will need to be deleted. Deletion can pose a problem for the CE Store because new statements are sometimes automatically inferred according to the rules already present in the store.

Although limited deletion of sentences is no obstacle, some kinds of changes to an ontology (that ought to be allowed) could potentially require identifying and deleting sentences en masse, at least if there are already statements about instances of the classes in the ontology. For example, we

[2] An alpha version is publically available for download at http://ibm.co/RDIa53.

might have a relation "receives information from" between artifacts, with some instances like "the machine KRF343 receives information from the machine EELR." and "the machine KRF343 receives information from the machine KRF343." Now suppose that someone alters the ontology by marking the "receives information from" relation as reflexive, which triggers (for every artifact X) the addition of a statement, "X receives information from X." Then, if the reflexivity of this relation is removed, the proper behavior would not be to have all such statements removed, but to remove only those that statements whose existence were generated by the rule. In that case, we might need to keep "the machine KRF343 receives information from the machine KRF343," because, say, it is a special radio that broadcasts messages and also records what it broadcasts in addition to what it receives from other broadcasters. The CE Store can keep track of which rules its sentences are derived from (as "rationale graphs"); even then, however, there remains a problem of managing which information *should* be deleted.

A second related worry is that the ontology editing software does not have exclusive control over the CE Store, nor can it verify ahead of time whether all the rules CE Store will be compatible with changes that a user makes. If a user accidentally adds a statement to the ontology which renders it inconsistent, it is possible that rules that trigger on the statements in the ontology will generate conflicting assertions, which can trigger further undesired effects that might be hard to predict and difficult to undo.

A third deficiency of the CE Store for maintaining the ontology information that users are editing is that it does not provide a means for keeping different users' ontology changes separate. In its current state, the CE Store does not allow any name spaces, which could differentiate between conflicting definitions of an entity.

Consequently, we found it advisable to adopt two design policies. First, it should not be required for the CE Store to ingest the ontology information, which is already in the OWL file. Second, we should not require the content of the OWL file match what ontology information (if any) kept in the CE Store. At the same time, we determined that we could still use the CE Store by inserting ITA-CE statements about the user's intention to make an ontology change or about what information about the ontology the user would like to obtain.

The structure of the resultant application is such that a user types ordinary English sentences into either a terminal window or the conversational interface of the CE Store, which triggers a response from a specially designed Java class, OntologyAgent. This Java class parses English input in attempt to infer the user's intentions regarding the loaded ontology. In the special case where the user's input is in controlled English, the OntologyAgent can act on it without needing to ask the user for further clarification, but when the user states a command in ordinary English, the OntologyAgent will do its best to interpret the input, sometimes responding with advice or questions. The output to the user is often in ordinary English, but when the OntologyAgent is prompting the user for confirmation, it provides a controlled English statement to coach the user so that on future occasions, the user can use controlled English for quicker unambiguous communication.

When the OntologyAgent needs information about the current state of the ontology, it sends out requests for information by placing appropriately structured ITA-CE sentences in the CE Store, which in turn trigger a response by another specially designed Java class, OwlAgent, which loads OWL ontology files and operates automated reasoners to answer questions about the inferred ontology, sending answers back to the OntologyAgent through ITA-CE statements placed in the CE Store. Information is shuttled back and forth between the OwlAgent and OntologyAgent until the OntologyAgent feels confident about the meaning of the user's initial request. Once the user has confirmed the accuracy of the OntologyAgent's interpretation, it passes an ITA-CE string to an OwlAgent *via* the CE Store conversational interface, the OwlAgent analyzes the requested modification, and, so long as it protects the integrity of the OWL file, updates the ontology accordingly. Given this structure, the program can (and does) log changes to all ontologies, keeps different users' ontologies separate, and saves changes to the ontologies incrementally to allow users to undo changes.

In order to enhance its interpretive capabilities, the OntologyAgent attempts to leverage information already present within the OWL file. It does so by posing queries to the OwlAgent (again, *via* the CE Store), such as, 'Does the class C exist in the ontology?' and 'Does the user's request violate any domain or range restrictions on object properties?' The OwlAgent, who manages changes to OWL files, answers the OntologyAgent's queries, for example, 'There is no such class in the OWL file' or 'That requested change would violate a domain restriction'. As a result, the OntologyAgent can make suggestions based on the information already contained within the OWL file and thus provide guidance to users wishing to modify an ontology. Currently, the software allows users to add classes anywhere in the class hierarchy and to add any desired existential restrictions. With the basic framework having been coded, it is straightforward to expand the software to allow other types of ontology changes.

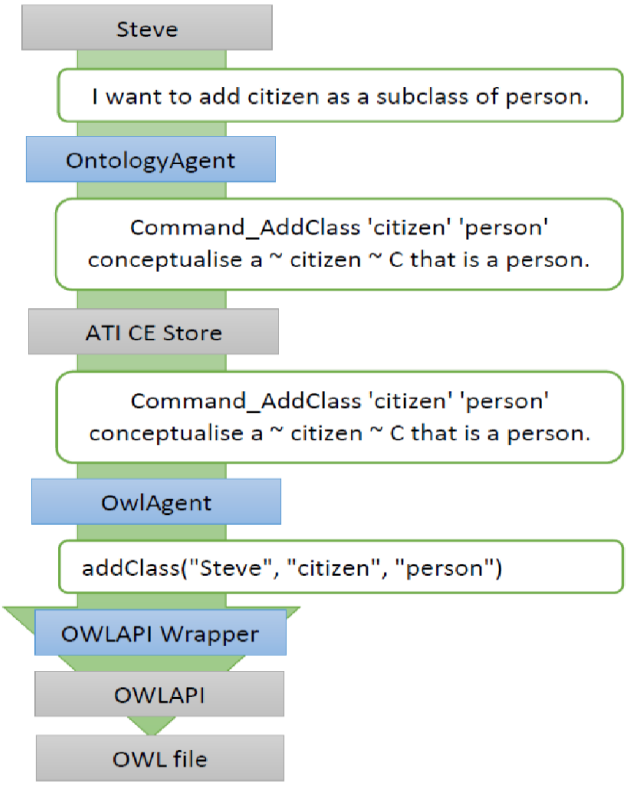
Consider an example. The user, who is logged in as 'Steve', types "I want to add Citizen as a subclass of Person." into the conversational interface. The OntologyAgent is part of

the CUBRC software and conducts some basic natural language processing to identify that the string matches one of the allowed forms for stating that that one class X should be made a subclass of another class Y. The user could have just typed "citizen is person" and gotten the same result. The interface is designed to be very forgiving about what it accepts. After getting confirmation from the user, the

OntologyAgent passes the user's command into the CE Store using a controlled English statement

```
there is a request card named 'msg_1' that is from
the service 'OntologyAgent' and is to the
individual 'OwlAgent' and has and has 'citizen' as
argument one and has 'person' as argument two and
has 'Steve' as human and has 'Command_AddClass' as
command and has 'conceptualise a ~ citizen ~ C
that is a person.' as content.
```

This string, when it enters the CE Store is recognized as a



legitimate CE statement. Because it is recognized as a card to the individual 'OwlAgent', the OwlAgent coded as part of the CUBRC software is passed the string through a Java method. The OwlAgent then parses the string to find that it should check whether 'citizen' can be added as a subclass of 'person'. It does so by making calls to a layer of the CUBRC software that keeps track of which ontologies are being edited by which humans and limits the kind of ontology changes permitted. This OWLAPI wrapper then makes calls to a publicly available Java API called OWLAPI, which is more complex and permits a much wider range of operations on ontologies, especially those stored as OWL files.

Note, however, that the OntologyAgent sends redundant information to the OwlAgent. In the content string it sends a statement that obeys the syntax and semantics of ITA-CE:

```
conceptualise a ~ citizen ~ C that is a person.
```

In the command, argument one, and argument two strings, it conveys in effect the same information:

```
Command_AddClass 'Citizen' 'Person'
```

The reason for passing two strings to the OwlAgent that express the same command is that (1) it is easy for the OntologyAgent to generate a tiny amount of additional text to express the same command in different formats, and (2) any agents that are coded to receive commands from an OntologyAgent can be coded to respond to whichever format is easiest to parse. In our conversational interface software, the Java methods called by the OwlAgent correspond exactly with the command strings that the OntologyAgent sends to the OwlAgent. The correspondence exists because there are only a small number of types of ontology alterations that it is reasonable to allow. Because OntologyAgents and OwlAgents understand that adding class X as a subclass of class Y is something a user probably wants to do, an OntologyAgent could encode that command in ITA-CE and have an OwlAgent decode it, but it is a bit simpler for the CUBRC software (and no harder for the CE Store) to have the OwlAgent ignore the ITA-CE altogether and check the command string and its arguments directly.

This raises interesting questions about what role, if any, controlled English can or should play in a conversational interface for editing ontologies. In the current version of the software, ITA-CE strings are used to provide a logically unambiguous yet human-readable confirmation prompt, but are not needed for processing the ordinary English input. Three questions in particular arise. Should some CNL be used even for the confirmation prompt? Should some CNL be used in this software for transmitting messages between the OntologyAgent and OwlAgent? And is ITA-CE the best form of CE to be used if we do use CE for user confirmation?

Concerning the first, the simplicity and lack of ambiguity provides a reason to use it for a command confirmation. But this alone is not a strong reason because it is often possible to make CNL statements even clearer in meaning by dropping some of the formality. For example, instead of the formula in ITA-CE:

```
conceptualise a ~ citizen ~ C that is a person.
```

the computer could offer

```
conceptualise a citizen that is a person.
```

without any loss of content. The tildes and the variables just indicate which concept is being defined, which is often clear from context or can be emphasized with boldface if desired. A stronger reason for using CNL is that it helps to coach the user so that the next time the user wants to enter the same kind of request, the user can use the sort of phrasing that appeared in previous prompts.

Concerning the second question of whether CNL should be used for the transmission of messages, there are some good arguments for its use. In our particular software, we controlled both ends: the natural language processing and the ontology processing. But if we think more broadly about how the conversational interface could be used, we find that an OntologyAgent could potentially get clues about how best to advise the human user from a wide range of

additional sources beyond what is kept in the edited ontology. Given the existing software framework, one could code agents that respond to an OntologyAgent's information requests by looking at a relational database, dictionaries, or even special purpose ontology advisors. Or one could code agents that edit ontologies stored in Open Biological and Biomedical Ontology (OBO) files rather than OWL files. The separation of the functions requires some sort of common communication format. Because CNL is relatively easy to parse and easy to extend in order to add additional functionality, it could serve well in this role. A further benefit of allowing CNL to play this role would be that one could easily set up logging and auditing software to work with the CE Store in order to keep track of which ontology changes were being made and by whom. Such information could be handy for later analysis to ascertain common ontology editing patterns, which could then be used to automate revisions to ontologies.

Concerning the third question, it is important to keep in mind that ITA-CE is severely restricted in its core lexicon and syntax. For example, the words 'every', 'some', 'or', and 'not' are not considered grammatical. While these same ideas can be expressed indirectly, the resultant statements and rules necessary for that expression tend to be too awkward for casual users (e.g., 'or' is defined in terms of complex statements involving numerous cases of 'and' and 'it is false that'). Thus, extending the lexicon and syntax of ITA-CE would greatly facilitate user understanding if these terms could appear in the confirmation prompts and in any communication to reduce complexity. In this way, ITA-CE could emulate some of the existing CNL-based ontology editors mentioned above, insofar as these editors utilize CNL constructs that more closely resemble OWL constructs.

V. USE CASES: MODIFYING THE SENSOR ONTOLOGY

A. *Adding New Classes*

It is easy to add a new class to an ontology using the conversational interface. The easiest way to do this is for the user to use the "is a" form that is standard in Basic Formal Ontology. The user types

```
An animal is an object.
```

At this point, the OntologyAgent sends a test message to the OwlAgent to see if the class "animal" can be assigned as a direct subclass of "object". If it can, the OwlAgent lets the OntologyAgent know this, and the OntologyAgent asks the user for confirmation. If the user answers, "yes", the OntologyAgent tells the OwlAgent to add the appropriate axiom and the OwlAgent does so, logging the change in case the user later wants to undo it. If the user answers "no" or doesn't respond or issues some alternative statement, The OntologyAgent will forget about the attempt to add "animal".

The user has other ways to communicate the desire to add a class. The following statements are equivalent to "An animal is an object."

```
animal is object
Animals are objects.
I would like to make animal a subclass of object.
Make animal a kind of object.
```

The software also allows users to insert a new class in between two existing classes, one of which is a direct subclass of the other.

```
Insert the class organism between object and
animal.
```

B. *Adding Existential Restrictions*

The other main capability of the software allows the user to add a new existential restriction on a class. The user can type something like

```
Every sensor observation is about some detected
material entity.
```
or
```
Sensor observations are about detected material
entities.
```

When receiving such a statement, the OntologyAgent sends a test message to the OwlAgent to see if the class "sensor observation" can have an existential restriction with the relation "is about" and the class "detected material entity". If it can, the OwlAgent lets the OntologyAgent know this, and then the OntologyAgent asks the user for confirmation. If the user answers, "yes", the OntologyAgent tells the OwlAgent to add the appropriate axiom and the OwlAgent does so, logging the change in case the user later wants to undo it. If the user answers "no" or doesn't respond or issues some alternative statement, the OntologyAgent will forget about the attempt to add the existential restriction.

C. *Adding Existential Restrictions in the Face of Obstacles*

In order to demonstrate some of the more sophisticated capabilities of our application, we created a use case with the following vignette:

> Your current version of the CUBRC Sensor Ontology includes a class called 'Detected Material Entity' but does not include any additional information about it.
>
> You want to introduce a class called 'Sensor Observation' and make the ontology understand that a sensor observation is the kind of thing that is about detected material entities. The ontology already has an 'is about' relationship but does not know the term 'Sensor Observation'.
>
> Your goals, then, are (1) to add a new class for sensor observations, (2), correctly situate that class within the existent hierarchy of classes, and (3) describe its relationship to detected material entities.

This task is more complicated than simply adding an existential restriction, but nonetheless quickly doable for the user who interacts with the OntologyAgent. The process

begins when the user enters a natural language expression that informally captures his request to add an existential restriction. The user could type anything like the following:

```
I want to make every sensor observation be about
some detected material entity.
I want to make sensor observation be about
detected material entities.
I want sensor observations to be about detected
material entities.
Sensor observations should be about detected
material entities.
Sensor observations are about detected material
entities.
a sensor observation is about a detected material
entity.
```

With such a request, the OntologyAgent queries for the class 'sensor observation'. Discovering – as the vignette stipulates – that there is no class by that name in the ontology, the OntologyAgent begins to search for any clues that will help interpret the user's intention. The OntologyAgent first checks the various parts of the user's phrase. In this case, it sees that 'observation' is a word in 'sensor observation' and checks whether there is a class 'observation' that could be a superclass for 'sensor observation'. When the OwlAgent tells the OntologyAgent that 'observation' is also not in the ontology, the OntologyAgent looks for restrictions upon relations (e.g., domain or range restrictions) in this case, whether the 'is about' relation already has any existential restrictions. (Although we have not coded further capabilities into the OntologyAgent yet, it would be easy to extend the CUBRC software to also look for synonyms of class names and annotations like developer comments, definitions, and class labels.) When searching the 'is about' relation for restrictions, the OwlAgent reports to the OntologyAgent the following clue: there is an axiom about the class *Information Content Entity* to the effect that *Information Content Entity* is equivalent to 'is about some *entity*' (a generic class comprising all other classes). Since the user is trying to express that sensor observations are about detected material entities, and since being an *Information Content Entity* is equivalent to being about some *Entity*, the OntologyAgent formulates a guess that (a) we need to insert a new class called *Sensor Observation*, and (b) the class *Sensor Observation* needs to be a subclass of *Information Content Entity*, though not necessarily a direct subclass. So, the OntologyAgent gets the direct subclasses of *Information Content Entity* from the OwlAgent and presents to the user the following text.

```
OntologyAgent: I understand you as wanting to add
an existential restriction. You are trying to
express that every sensor observation is about
some detected material entity. I think you want me
to issue the following command:
conceptualise a ~ sensor observation ~ C that is
about the detected material entity R.
However, the ontology is not aware of what 'sensor
observation' means. Because of the 'is about'
relation, a sensor observation should probably be
assigned to be one of the following:
1. A descriptive information content entity
2. A designative information content entity
3. A directive information content entity
4. An information content entity
If is one of these, just let me know the number,
or you can write something of the form, "a sensor
observation is a ___."
```

As it turns out, the user Steve decides to situate *Sensor Observation* under the parent class of *Descriptive ICE*, so he types

```
1
```

The OntologyAgent responds by adding 'sensor observation' as a direct subclass of the 'descriptive information content entity' class. Once the OwlAgent reports that that change has been made, the OntologyAgent knows that the obstacles have been cleared to allow the existential restriction to be added. So the OntologyAgent immediately sends the command to the OwlAgent without needing further confirmation. When the OwlAgent reports that this further command has succeeded, the OntologyAgent reports to the user the following:

```
OntologyAgent: I successfully added 'sensor
observation' as a direct subclass of 'descriptive
information content entity'. I successfully added
the existential restriction you requested.
```

In summary, Steve submitted a single sentence to convey his desired ontology change. After that, Steve only needed to select a multiple choice answer in order to clarify where the previously unknown 'sensor observation' class should be placed in the hierarchy, and all the necessary changes were made without any further effort.

In practice, all ontology changes would be audited before being integrated into widely shared OWL files.

VI. FUTURE WORK

Our research on this topic was merely exploratory because of resource limitation and a number of future efforts are warranted. Greater editing functionality should be added first because it provides clear value and is not an ambitious task. The harder task will be to improve the software's ability to help the user with the most appropriate questions and advice. Still, even these improvements can be incremental and piece-meal, as the natural language is meant only to help guide and coach when the user does not know the appropriate controlled English to use.

Earlier, we showed that although our application extends the capability of the CE Store software, it does so without necessarily using ITA-CE to represent ontological claims. The role played by ITA-CE might easily be played by a more expressive CNL or a standard data format like XML or JSON. There is potential for ITA-CE to serve as a common format for capturing the semantic intention of natural language user inputs. Its value in this regard will largely depend on how much other infrastructure uses ITA-CE and how well the information being passed back and forth can be leveraged to automate some ontology development.

At this stage, our application is limited to assistance with OWL-based ontology editing. It is our hope that this project will eventually be extended further, so that ITA-CE

sentences can be exploited to mediate user interaction with other ontology languages (e.g., OBO) and other formats (SQL, relational databases). Thus, in dividing labor between the OntologyAgent and OwlAgent, we have left open the possibility of mapping ITA-CE to these formats and of writing additional back-end agents, whose jobs, like the OwlAgent's, would be to exchange information with the OntologyAgent and edit the appropriate document on behalf of the user. We also leave open the possibility of alternative OntologyAgents, who pass messages in ITA-CE only.

To illustrate this proposal, consider the following two scenarios. In *Scenario #1*, a user interacts with a different OntologyAgent that outputs ITA-CE strings only, not the command strings described earlier, and sends the messages to an OwlAgent. The problem is that the OwlAgent does not understand ITA-CE strings; it ingests command strings only. In this scenario, it would be desirable to introduce a further intermediary agent to translate ITA-CE strings into command strings amenable to the OwlAgent's work. By introducing this further layer, we allow the OwlAgent to continue interpreting messages in terms of command strings. At the same time, this makes room for ITA-CE to act as middleware for various other agents.

In *Scenario #2*, a user interacts with the OntologyAgent, who passes messages in turn to an OwlAgent that understands ITA-CE strings only, not command strings. This new OwlAgent would have access to a relational database and would query that database to provide the OntologyAgent with information about the user's intended request. In this scenario, it becomes indispensable that the OntologyAgent transmit not just command strings, but ITA-CE strings as well. Thus, if ITA-CE is to be the common format for various agents and information sources, we ought to ensure that OntologyAgents are conversant in ITA-CE.

These scenarios highlight another potential benefit of employing ITA-CE as middleware. If ITA-CE is used as a common format, then additional agents could be programmed to provide OntologyAgents with answers to his queries which are based on the access they enjoy to various information sources. In this case, ITA-CE appears to be promising as a common format for the exchange of information among agents.

If ITA-CE were to be harnessed in these ways, then in will be necessary to augment the present ITA-CE core lexicon and syntax. At present, ITA-CE does not allow the terms 'every', 'some', 'or', and 'not', which inhibits users from facile comprehension. Thus, an extension of this project would be to pursue methods laid out by Mott and Hendler [2], in which new layers of generic syntax are added to the core ITA-CE syntax. Mott and Hendler illustrate this with the adverb 'only'; to add 'only' to the syntax of ITA-CE, they invented a language, which defines 'only' in terms of the unaugmented core syntax. Further transformations could add common quantifiers and connectives such as 'every', 'some', 'or', and 'not', thus enhancing users' interaction with ITA-CE expressions by rendering them more natural.

Acknowledgments

Work on ITA-CE as a middleware for ontology editing was supported by the US Army Research Laboratory. We also thank David Mott for his consultations on ITA-CE.

References

[1] Bock, C., Fokoure, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M. 2012. "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax," 2nd

[2] Mott, D. Hendler, J. 2009. "Layered Controlled Natural Languages," *Proceedings of the Third Annual Conference of the International Technology Alliance*.

[3] Braines, D., Mott, D., Laws, S., de Mel, G., Pham, T. 2013. "Controlled English to Facilitate Human/Machine Analytical Processing," in Proceedings of SPIE.

[4] Braines, D., Preece, A., de Mel, G., Pham T. 2014. "Enabling CoIST Users: D2D at the Network Edge," in 2014 Proceedings of the 17th International Conference on Information Fusion (FUSION).

[5] Poteet, S., Xue, P., Kao, A., Mott, D., Braines, D., Giammanco, C. 2013. "Controlled English for Effective Communication during Coalition Operations," Proceedings of ICCRTS.

[6] Preece, A., Pizzocaro, D., Braines, D., Mott, D., de Mel, G., Pham, T. 2012. "Integrating Hard and Soft Information Sources for D2D using Controlled Natural Languages," *Proceedings of the 15th International Conference on Information Fusion*.

[7] Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang H., Wroe, C. 2004. "OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns," *European Conference on Knowledge Acquisition* (EKAW-2004), Whittlebury, UK.

[8] Bernstein, A., Kaufmann E. 2006. "GINO – A Guided Input Natural Language Ontology Editor," in Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.), ISWC 2006. *Lecture Notes on Computer Science*, Vol 4273.

[9] Bernstein, A., Kaufmann, E., Kiefer, C. 2009. "Querying the Semantic Web with Ginseng – A Guided Input Natural Language Search Engine," *Searching Answers: Festschrift in Honour of Michael Hess on the Occasion of His 60th Birthday*, Clematide, S., Klenner, M., Volk, M. (eds.). Munster: MV-Wissenschaft.

[10] Kaufmann, E., Bernstein, A. 2007. "How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users?" *Lecture Notes in Computer Science*, Vol. 4825, 281-294.

[11] Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S. 2007. "CLOnE: Controlled Language for Ontology Editing," *Lecture Notes in Computer Science: The Semantic Web*

[12] Hart, G., Johnson, M., Dolbear, C. 2008. "Rabbit: Developing a Controlled Natural Language for Authoring Ontologies," *The Semantic Web: Research and Applications, Lecture Notes in Computer Science*, Vol. 5021, 348-360.

[13] Kaljurand, K. 2008. "ACE View – An Ontology and Rule Editor Based on Controlled English," *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference* (ISWC 2008), CUER Workshop Proceedings.

[14] Wroblewska, A., Kaplanski, P., Zarzycki, P., Lugowska, I. 2013. "Semantic Rules Representation in Controlled Natural Language in FluentEditor," *The 6th Annual International Conference on Human System Interaction* (HSI).

[15] Smart, P.R. 2008. "Controlled Natural Languages and the Semantic Web. Technical Report ITA/P12/SemWebCNL, School of Electronics and Computer Science, University of Southampton.

[16] Horridge, M., & Patel-Schneider, P.F. 2012. "OWL 2 Web Ontology Language Manchester Syntax," 2nd edition.

[17] Cregan, A., Schwitter, R., Meyer T. 2007. "Sydney OWL Syntax – toward a Controlled Natural Language Syntax for OWL 1.1," presented at *OWLED 2007, OWL: Experiences and Directions, Third International Workshop*, Innsbruck, Austria, 6-7th, June 2007.

[18] Sowa, J.F. 2007. "Common Logic Controlled English," http://www.jfsowa.com/clce/clce07.htm.

[19] Mott, D. 2010. "Summary of ITA Controlled English," https://www.usukita.org/papers/5658/details.html.