Software Quality Journal (2022) 30:227–251
https://doi.org/10.1007/s11219-020-09544-9

Ontology-based metamorphic testing for chatbots

Josip Božić[1]

Accepted: 16 December 2020 / Published online: 27 April 2021

Abstract
Modern-day demands for services often require an availability on a 24/7 basis as well as online accessibility around the globe. For this sake, personalized virtual assistants, called chatbots, are implemented. Such systems offer services, goods or information in natural language. These natural language processing (NLP) programs respond to the user in real time and offer an intuitive and simple interface to interact with. Advantages like these make them increasingly popular. Therefore, ensuring correct functionality of chatbots is of increasing importance. However, since different implementations and user behaviour result in unpredictable results, the chatbot's input and output data are difficult to predict and classify as well. Under such circumstances, test cases can be inferred from the domain of possible inputs of a system under test (SUT). Ontologies are concepts used in AI to provide formal representations of knowledge for a specific domain. Such ontological models contain structured information that is used for test generation. On the other hand, testing of chatbots represents a challenge because of the absence of a test oracle. In this paper, both challenges are addressed by conceptualizing ontologies for input generation and output processing in form of a metamorphic testing approach. In this scenario, both concepts are applied for automated testing of chatbots. The approach is demonstrated on a real system from the tourism domain, thereby discussing the obtained results.

Keywords Ontologies · Metamorphic testing · Functional testing · Chatbots

1 Introduction

Natural language processing (NLP) is a discipline that started around 1950 with the introduction of the famous Turing test Turing (1950). Virtual assistants are programs that realize the communication with a user in natural language. These NLP programs, called chatbots Mauldin (1994), offer the advantage to resemble a natural and intuitive way of interaction. Usually, these programs comprehend information from a certain domain. In such way, the chatbot provides specific information in an often entertaining and anonymous manner Brandtzæg and Følstad (2017). Since several studies predict the rise of the chatbot market in the future, addressing the functionality of these systems becomes essential

✉ Josip Božić
jbozic@ist.tugraz.at

[1] Institute of Software Technology, Graz University of Technology, Graz A-8010, Austria

Følstad and Brandtzæg (2017); Grudin (2019). Until now, only a few testing approaches exist that check the correctness of chatbots (e.g. Vasconcelos et al. (2017); Bozic (2019)). However, since a user can address a chatbot in different ways, the input scope might be difficult to predict. In additional to that, testing the chatbot in a generalized way proves problematic due to the absence of expected values.

The first issue is addressed by considering to formalize information from the SUT's domain. For this reason, the notion of ontologies is applied for testing purposes. Originally, an ontology is a concept that comes from philosophy and discusses the nature of being Oxford Reference (2020). However, this concept has been developed further to describe formalizations of knowledge from a specific domain Gruber (1993); Guarino and Musen (2005). The general benefit of ontologies is that they provide a structure of reusable information for a specific purpose. Since they define restrictions and relations between components, they allow to generate reasonable outputs Noy and McGuinness (2001). Also, they can be extended with additional information without affecting the overall structure. In this paper, the ontology represents a model that conceptualizes domain-dependent knowledge for NLP. In this case, an ontology represents a comprehensive input model that supplements the generation of natural language. In fact, a system of words and relations comprises a linguistic knowledge base in form of an ontology model. The goal of such system is the generation of sentences, i.e. test cases, that check for functional correctness of a NLP system. However, since ontologies serve as input models, they have to be supplemented by other components in order to fulfil a purpose.

Chatbots are programs that come in different flavours for multiple purposes. Because of this, their behaviour might be difficult to predict. This means that traditional testing approaches might not always provide an optimal solution. In testing, it's common practice to rely on test oracles in order to obtain test verdicts. However, problems may arise due to nontestable programs or because of the high amount of manual effort needed to draw a test conclusion Weyuker (1982). This can be due to multiple reasons, like high costs or complexity of implementations or the domain. Driven by the same issue, the concept of *metamorphic testing* (MT) has been introduced in Chen et al. (1998). In MT, the output of a system is not checked against an expected value but a *metamorphic relation* (MR). These describe a relation between the input data and the corresponding output. New test cases are inferred from previous ones according to the specified MRs. These inferred test cases are meant to detect errors that the former ones failed to do. MT is not meant only to detect errors in cases where no expected values are defined, but helps to reveal errors in the production phase as well. In addition to that, MT does not depend on a specific testing strategy; thus, it can be combined with other test selection approaches (see Section 7). In fact, dealing with such situations, where no test oracle is available during testing, represents the second issue of this paper.

This paper represents an extension of the previous work in Bozic and Wotawa (2019), which introduced a metamorphic testing approach for chatbots. As in the former case, this work continues to focus on chatbots that use textual communication. Additionally, the current work introduces ontologies as the basis for data conceptualization and test generation. This novel approach is defined in terms of ontologies and MT and checks the functionality of a chatbot in an automated manner. Basically, the motivation behind this approach is to guarantee functional correctness of a system under the absence of expected data. The theoretical background is provided, and the implementation is elaborated on a chatbot from the tourism industry.

The remainder of the paper is structured as follows: First, Section 2 introduces a chatbot from the tourism domain that serves as the SUT in this paper. Then, Section 3 introduces

basic concepts of ontologies for this NLP system. Then, Section 4 introduces metamorphic testing and its adaptation to testing of chatbots. Section 5 describes the proposed ontology-based MT approach for chatbots. The implementation is evaluated in Section 6, and related work is enumerated in Section 7. Finally, Section 8 concludes the work.

2 Testing of a NLP system

As already mentioned, the proposed MT approach for testing of chatbots is already described and evaluated in Bozic and Wotawa (2019) on a hotel booking system. In this paper, technical details about the implementation are omitted in favour of providing a few details about its functionality. The virtual agent is implemented in Dialogflow Diagflow (2018) and communicates with the user by sending and receiving messages in natural language. Since the goal of the chatbot is to make a hotel booking, it demands some information in order to finalize a reservation. On the side of the chatbot, this information is hard coded in form of eight mandatory parameters. During communication, the agent will ask the user to provide this information. In turn, it's assumed that the chatbot does NLP and comprises necessary data. A typical communication flow, initiated by the user, does look as follows.

```
I want to book a room in Madrid.
- Do you want a hotel, hostel or apartment?
book a hotel
- Which one do you prefer?
Elmhurst
- When do you want to start your stay?
I want to check-in today.
- How many adults?
2 adults
- How many children?
2 children
- How many nights?
1 night
- How many stars do you prefer?
i want 5 stars
- Let me sum up! Location: Madrid, Name: Elmhurst, Check-In:
  2019-05-20, Accommodation: hotel, People: 2 adults, 2 children,
  Nights: 1, Stars: 5 stars
```

In case that the user does not provide a specific information, the chatbot will continue to ask for clarification. Thus, no reservation will be made. On the other hand, the behaviour of the user can be unpredictable to a certain degree as well. For example, the user can make an inquiry by rephrasing sentences or providing multiple information in one message. This fact will play a role in the test generation process. The chatbot is expected to recognize different types of input utterances. In addition to that, the chatbot understands additional messages, like exit or abort, but these represent rather control messages. Thus, they are not considered for test generation. However, regardless of the user input, the chatbot demands information about the following parameters.

$location, $venue-type, $venue-title, $checkin-date, $adult, $child, $night, $star

For the reservation to be completed, all these mandatory values must be provided by the user. If this requirement is not met, then the chatbot sends additional inquiries to the user. In fact, the agent demands clarification for missing information in natural language. The communication loop will get stuck as long as no information is provided for the mentioned keywords. A reservation is finalized after all required data are obtained, thereby returning a confirmation message. In theory, a general testing approach that could be defined to every chatbot from the same domain. In such case, it can be assumed that at least some of the above information is required for every hotel booking system.

Since the user provides information in an optional manner, there is no proscribed order or form of inputs. Since our motivation is to automatically generate and execute user inputs, an initial data conceptualization is needed. The next section gives an overview about such a model that is used to test the aforementioned chatbot.

3 Ontologies for chatbot testing

As originally stated, ontologies represent formalized knowledge in form of a model Gruber (1993). The term itself, however, is influenced by Genesereth and Nilsson (1987) and conforms to *conceptual model* or *conceptualization*. Ontologies deal with entities that can be grouped within a hierarchy. Such entities are categorized in terms of concepts, relations, functions and instances. In fact, the resulting ontology model can be seen as a meta-model of a specific problem. The basic definition of ontologies, by following Gruber (1993); Gómez-Pérez (1999), can be stated as follows.

Definition 1 An ontology is defined with a tuple (C, R, A, I), where C denotes the set of classes (also called concepts) that define an abstract or real entity. R specifies the type of relationship between individual classes. These can be defined as a subset of a product of n sets; thus, $R : c_1 \times c_2 \times ... \times c_n$. A denotes a set of attributes where $a \in A$ represents one property of a class $c \in C$. The set I defines concrete instances of classes (also called individuals).

In this paper, ontologies are applied to conceptualize the scope of natural language for the purpose of generating comprehensive user inputs. Since the work deals with a hotel booking system, the domain comes from the tourism industry. Thus, the ontology contains information that can be used in order to make a valid hotel reservation. The idea is to emulate an interaction between a possible client and the booking system (see Section 2). Currently, several languages exist for specifying ontologies (e.g. OWL Web Ontology Language (2019), CycL Lenat et al. (1985), etc.) that rely on different strategies for specific purposes. In this paper, the common editor Protégé Musen (2015) is used for defining a OWL-based ontology model. OWL is an ontology language that describes a domain by applying first-order logic rules. It is based on formal semantics of RDF 1.1 (2020) but extends its syntax with additional linguistic capabilities.

The goal of the presented ontology is to encompass structured linguistic information in a way that it serves as a basis for language generation. Natural language, such as English, can be defined as a sequence or set of sentences. Each sentence is defined by a set of words, which occur in groups that rely on a specific syntax. For example, groups can be

constituted in sentences in form of noun phrases (NP) or verb phrases (VP). Subsequently, these groups can be divided further into word classes, such as pronouns (PN), nouns (NN), verbs (V), adverbs (AD), determiners (D), prepositions (PR) and the special case of *to-infinitives* (TI). For this matter, let's analyse the following user inquiries with regard to their word classes:

(a) "I want to check-in today"
(b) "book a hostel"
(c) "I want to book a hotel in Madrid"

As can be seen, the first and last sentences comprise a sentence that starts with a personal pronoun. The second one reflects situations where a user simplifies the demand. It represents a VP that is followed by one argument. On the other hand, the third sentence contains reoccurring word classes. Basically, the sentence is structured by several subsequent NPs and VPs.

In order to provide an illustration, let's depict these sentences in form of syntax trees, as given in Figs. 1, 2 and 3. In all three cases, the sentences are organized in a constituency structure. Every parse tree consists of individual elements, like word classes and phrases, which represent a sentence constituent. Phrases are divided further into subsequent words that are semantically connected according to the tree's branches. The first two cases represent simple user inputs with a comprehensible small number of constituents. However, in the third case, verbs are lined up as parts of a deeper tree structure. Since a user provides inputs in an optional manner, the ontology model is meant to provide enough information for every dialogue scenario. Also, the tested chatbot is required to correctly interpret a variety of user inquiries, regardless of its complexity. This requirement is taken into consideration in the ontology drafting process.

In Protégé, several lexical categories are defined in terms of the aforementioned ontological elements. The main elements in ontologies are classes, which are also

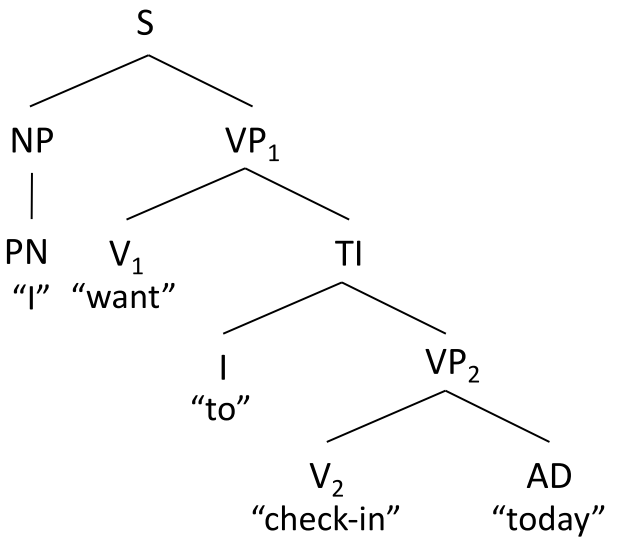
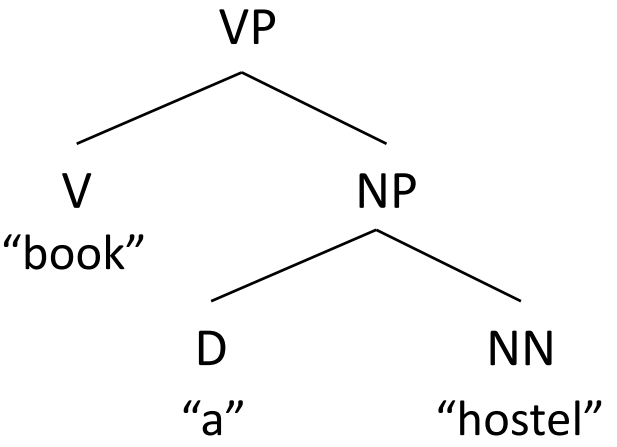


Fig. 1 Syntax trees for inquiry #1

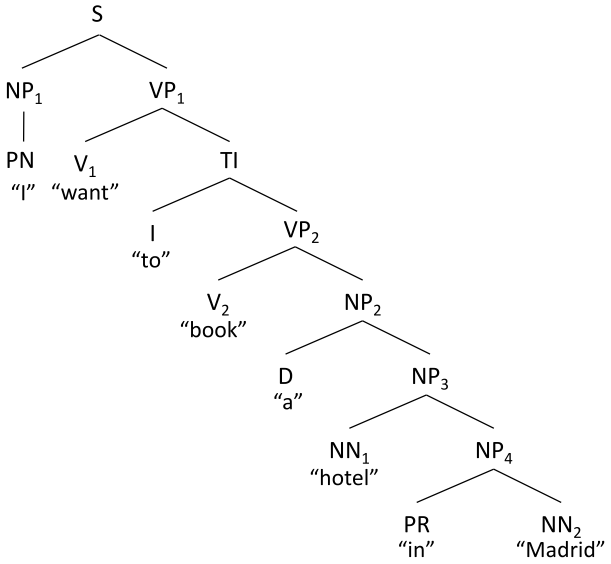
Fig. 2 Syntax trees for inquiry #2



called concepts. These are abstract elements that define a specific entity and their corresponding information. Object properties relate objects to other objects, i.e. describe relations between classes. Attributes, called data properties, characterize a concept by relating objects to datatype values. Finally, individuals comprise instantiated attributes that concretize class instances. The ontology model in this paper comprises one umbrella class and several subclasses. Concepts are defined in terms of abbreviated names of word classes. These concepts represent super-classes that are branched into several data properties. In turn, such data comprise a set of concrete values that can be assigned to a property. In accordance to the example from the above figures, a list of all data properties is summarized as follows:

- pronoun: Personal pronouns act as substitutes for nouns and denote the person of the client. In this paper, only the first person is used (“I”).
- noun: This element denotes the name of term that is used as part of a noun phrase. Usually, a noun comes in conjunction with other lexical items. However, it can occur in a stand-alone way in case of an answer to a chatbot’s inquiry.
- determiner: Determiners are divided into articles and quantifiers. In this case, the indefinite article usually precedes a noun and succeeds a verb. A quantifier defines numbers and quantify a noun.
- preposition: This word expresses a spatial relation between nouns. In such way, it only acts as an adjunct to the preceding noun.
- verb: A verb is a word that denotes an action in its infinitive form. Often, a verb constitutes a connection between other lexical words, for example nouns and particles of other verbs.
- adverb: This word or expression describes time and can occur in conjunction with a verb. They can supplement other words in order to indicate time in the client’s inquiry.
- to-infinitive: The word *to* supplements a verb in order to define an intention.

Fig. 3 Syntax trees for inquiry #3



As already mentioned in Section 2, a chatbot requires mandatory information for some parameters. In order to address this requirement, this key information is encoded inside the ontology model. In an ontology, an instance represents a data property with an assigned concrete value. Such model must depict all possible values that can be assigned to a concept's property. Table 1 depicts all ontology concepts with its corresponding data properties and individuals' values. The initial values in the ontology are set manually by consulting the supported library from Dialogflow. In fact, data properties in the ontology resemble entities in Dialogflow. Thus, initial values in the ontology correspond to concrete values from the chatbot's entities.

An outstanding element in ontologies represents the network of relations. Actually, words and syntactic rules are depicted by setting relationships in the model. Relations determine the way in which words are combined in a sentence. Classes can be related to each other on the basis of their subclasses and individuals. The definition of such bonds represents one of the main advantages of ontologies in comparison to other data conceptualizations, e.g. taxonomies. This paper defines a total of two relations:

- hasSubclass: This relation denotes all concepts that are part of the main concept, namely a sentence. It defines all of the mentioned word classes as parts of their linguistic super-class.
- is-defined-by: This relation enumerates all instances of a concept. As already mentioned, this includes data properties with concrete values. In this ontology, a concept comprises one or multiple instances.

Figure 4 depicts the overall ontology model with corresponding relations between different concepts. Note that property assertions from Table 1, although not visible in the model, are constituent elements of the ontology. The function of this ontology model is to serve as a source for test generation. Test cases are generated by following the ontology definitions, which are translated into dialogue scenarios. Each such scenario

Table 1 Ontology components for NLP

Class	Individual(s)	Data property assertions
PN	pnoun	"I"
NN	location	"Madrid", "Vienna", "Prague"
	venue-type	"hotel", "hostel", "apartment"
	venue-title	"Sacher", "Elmhurst", "Fairmont"
	space	"room"
	child	"children"
	adult	"adults"
	night	"nights"
	star	"stars"
D	article	"a"
PR	preposition	"in"
V	verb	"check-in", "book", "want"
AD	checkin-date	"today", "yesterday", "tomorrow"
TI	to-infinitive	"to"

resembles a textual user interaction with the chatbot. For this reason, the ontology is drafted to define different semantic interpretations of a sentence. Although ontologies exist that rely on external lexical sources, for example the semantic database WordNet WordNet (2020), this work encompasses a more comprehensible amount of information.

4 Metamorphic testing for chatbots

In the previous works in Bozic (2019), a functional testing approach for chatbots was introduced. The hotel booking chatbot is checked whether valid reservations are processed correctly. However, due to unpredictive behaviour of such systems, proving their correctness represents a challenge. For this reason, a new testing paradigm needs to be introduced. Since MT addresses situations where only sparse or no knowledge exist about a system, it fits well to address this problem.

First, let's denote the basic principles of MT, as initially described in Chen et al. (1998) and Segura et al. (2016). In contrast to other testing techniques, the basic principle for test generation differs in MT.

Definition 2 Let f be an implemented function of a program P. R_f defines a property that must be valid for all inputs $x_1, x_2, ..., x_n$ over f, and the corresponding outputs $f(x_1), f(x_2), ..., f(x_n)$. The relation R_f represents a *metamorphic relation* (MR) over the input-output pairs; thus, $R_f((x_1, f(x_1)), (x_2, f(x_2)), ..., (x_n, f(x_n)))$, where $n > 1$.

Since no expected test value is available, properties have to be defined that must hold for all test inputs against a program, regardless of their value. However, conditions can be specified as part of a MR that put restrictions on test cases Segura et al. (2016). A metamorphic relation can represent any suitable type of relation, such as equalities, properties, constraints etc. The function of MR is twofold: 1) it serves as a guideline for generation of test inputs from existing data and 2) acts as an output relation, which is checked after the test execution.

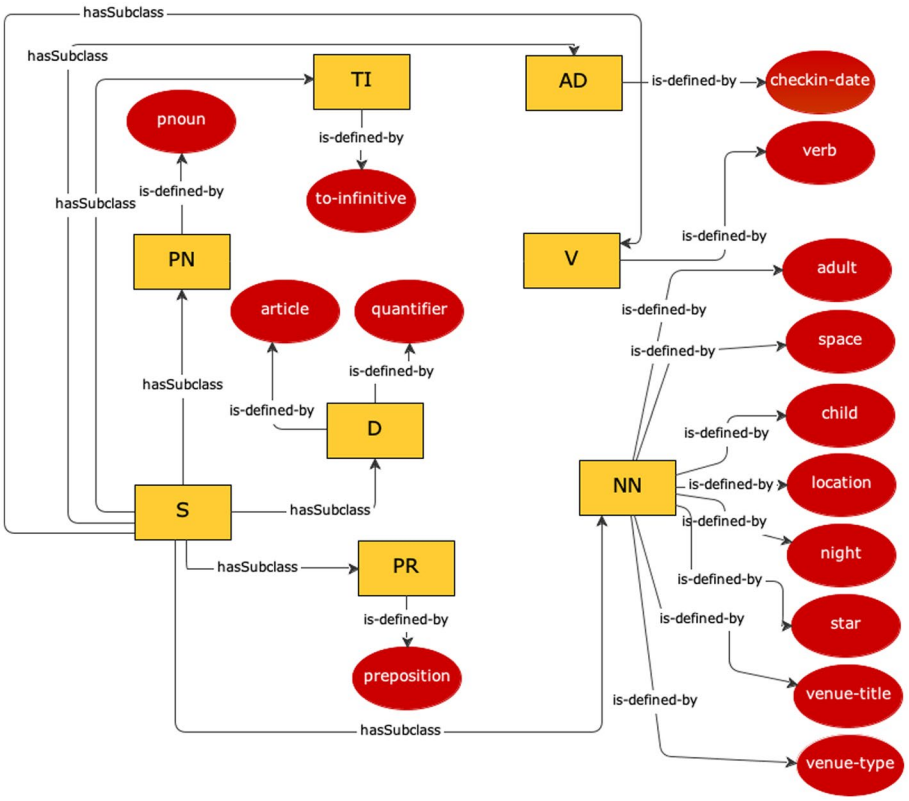


Fig. 4 Ontology for testing of chatbots

In such way, MR replaces the test oracle by comparing the input-output pairs from test cases. This comparison is denoted as a *metamorphic check* (MC). A metamorphic check fails in case that the input-output pair satisfies a metamorphic relation.

Definition 3 An initial source test case x_1 that does not trigger an error or vulnerability in P is a successful test case. A follow-up test case x_2 is inferred from x_1 according to a selection criteria, as specified by R_f. The pair of the source and follow-up test case can be defined as a *metamorphic test case*.

Usually, the source test case represents a random value or is derived from partial knowledge about the domain of P. All new test cases are derived from the successful test cases. For the source test case x_1 and its output $f(x_1)$, new test cases will be generated based on the input-output pair $(x_1, f(x_1))$. During test execution, the applied MRs must hold for input-output pairs of both, source and the follow-up, test cases. If the MR does not hold for a certain metamorphic test case Segura et al. (2016), then it's concluded that a fault or inconsistency is detected in P.

Now, let's define the following properties of the chatbot as MRs. According to classical MT, it's assumed that some information about the domain is known in advance Segura et al. (2018). In turn, this information is used in order to define MRs. As already mentioned, these relations must be valid for test inputs and outputs. It's required that a chatbot should be able to

understand a variety of different user inputs. In fact, a user is given the freedom to express her or his intention in different phrases. An intention must be recognized even by using different words. In addition to that, a chatbot must not derive false conclusions by misjudging user inputs. In order to challenge chatbot's NLP capabilities, it's tested by rephrasing user inputs. Lexical alternations, like paraphrases and substitutions are applied to keywords and submitted against the SUT. As already mentioned, MR affect only concrete values from the ontology, without addressing other elements. The source test case from Table 2 encompasses sentences in natural language. Several keywords are emphasized that will be addressed by the MRs. The MRs affect input generation for chatbots, as follows:

MR1: Replace keywords with synonyms: In order to infer I_f from I_s, the keywords are replaced by synonyms. Although, for every follow-up test case, keywords are changed in only one action. The obtained O_s serves as a reference that is compared to the obtained O_f. When applying synonyms, the same behaviour is expected from the chatbot; thus, $O_s = O_f$ must be valid.

MR2: Change numerical values: This relation instructs to change numerical values for individual actions. The outputs of O_s and O_f should be equal, with the exception of the numerals from the final reservation.

MR3: Omit words: This relation demands that keywords are omitted, thus providing insufficient information. The chatbot should recognize the difference, thus resulting in different responses. MR3 is valid in case that $O_s \neq O_f$ for every I_f.

MR4: Replace keywords with unrelated terms: This relation is similar to MR1 but instead of synonyms, keywords are exchanged with unrelated terms. Therefore, the corresponding outputs must never resemble each other. Therefore, the expected outputs equal $O_s \neq O_f$.

MR5: Change order of actions: The order of actions in I_s is changed in the follow-up test case I_f. Since the tester does not know about the SUT's behaviour, the obtained outputs are expected to differ; thus, $O_s \neq O_f$. (However, a valid reservation might still be accomplished accidentally by a valid random combination.)

The MRs and their correlation to test generation will be explained in detail in Section 5.2. In contrast to predecessor's work, metamorphic relations are applied to ontologies. In such way, ontologies can be compared with regard to concept descriptions. The next section discusses the relation in which the application of MRs to ontologies affects the test generation process.

Table 2 Source test case

#	Action	Parameter
0:	I want to book a room in Madrid	$location
1:	book a hotel	$venue-type
2:	Elmhurst	$venue-title
3:	I want to check-in today.	$checkin-date
4:	2 adults	$adult
5:	2 children	$child
6:	1 night	$night
7:	i want 5 stars	$star

5 Ontology-based metamorphic testing

The combination between ontologies and MT can be defined as follows. Metamorphosis is the process that transforms an abstract or concrete entity into a new entity. During this process, the subsequent entity keeps more or less common attributes with the original form. However, other attributes of the original entity are modified to a degree that they describe a new entity. In theory and practice, any system fulfils at least one goal function. A metamorphosis, i.e. a process that adds changes to the system, can be applied when the current system is not efficient enough to fulfil that function. In such case, the metamorphosis represents an optimization of the original entity.

For this reason, this paper suggests the combination of ontologies and MT. Here, MRs are applied to the initial ontology so that different models are obtained. That is, changes are added to original specification, thus resulting in a new model. In this case, changes represent mutation operations that are applied to all constituent elements of the ontology. It should be noted that the relations between ontology concepts will remain the same; however, underlying data assertions will change. In such way, different test cases are generated that eventually trigger incorrect behaviour where previous tests failed. Ultimately, the new ontology is compared to the preceding one with regard to achieving the testing goal.

The previous paper in Bozic and Wotawa (2019) addressed the mentioned challenges as part of the MT approach. Whereas, this work adds ontologies as its starting point. In general, the following tasks constitute the proposed ontology-based MT approach.

1. Grammar inference
2. Test generation
3. Test execution

A description is given how the ontology can be applied for testing purposes. First, a grammar-based test generation technique is elaborated. The MT-related tasks will be discussed afterwards in conjunction with the new approach. Finally, the test execution is explained by following the underlying algorithm. For all these cases, the adaptation of MT to chatbots has to be addressed in detail.

5.1 Grammar inference

In software testing, formal grammars are often used for test generation. This fact makes them suitable for adaptations to different testing methods. In addition to that, formal grammars are also used in order to specify natural languages. However, the problem such grammars is that they are effort-intensive to create and to maintain. This is especially the case when depicting information from larger domains Cimiano et al. (2014). Since grammars rely on domain-dependent information, an ontology can serve as the corresponding knowledge domain for this purpose. Subsequently, grammars can be inferred from ontologies, which eases their development process. A context-free grammar (CFG) defines production rules in a formal representation. In this paper, CFGs are applied in order to depict the syntax of natural language. They comprehend linguistic elements and their mutual relations for generating sentences. Based on Pereira et al. (2016), a CFG is defined in the following way.

Definition 4 A context-free grammar is given with a tuple (N, T, S, R), where N and T define the sets of disjoint nonterminal and terminal symbols, respectively. S denotes the start symbol,

whereas R represents production rules that define how symbols are substituted by new symbols. Each rule $r \in R$ has the form $LHS \rightarrow RHS$, where the left-hand side (LHS) consists of a onterminal and the right-hand side (RHS) of one or multiple terminal or nonterminal symbols.

In order to derive a grammar from an ontology, formal mapping rules need to be defined. These rules guide the transition of every element from the OWL-based model to a CFG. By following mapping rules, the grammar development process reflects the ontological structure. In general, nonterminals correspond to concepts, which include phrases and word classes, and individuals. Every subconcept is linked to the main concept S (= sentence) into a hierarchy by the same relation type. Each concept from the ontology represents an option in first CFG production. These inference rules, influenced by Pereira et al. (2016), are given as follows:

- Concepts = Symbols (nonterminal or terminal)
- Relations and data properties = Grammar productions

In order to map elements from ontologies to grammars, their content must be put into a comprehensive data structure. Symbolically, noun and verb phrases from the ontology can be formalized as triples. Whereas words from the ontology take the role of subjects and objects, relations act as predicates.

$$(\text{subject},\ \text{predicate},\ \text{object})$$

For example, an inferred grammar production from an abstract example is given below. The example depicts two concepts from an ontology and a is-a relation.

$$(\text{B},\ \text{is-a},\ \text{A})\ \text{and}\ (\text{C},\ \text{is-a},\ \text{A}) \Rightarrow \text{A} \rightarrow \text{B} \mid \text{C}$$

The production rule determines that A can be substituted by B or C, regardless of the context of A. In the next examples, two triples from the ontology are converted into a CFG. The LHS constitutes a nonterminal, whereas the RHS depicts a disjunction between two nonterminals. On the other hand, relations are always terminals and parts of the RHS.

$$(\text{S},\ \text{hasSubclass},\ \text{PN})\ \text{and}\ (\text{S},\ \text{hasSubclass},\ \text{V}) \Rightarrow \text{S} \rightarrow \text{PN} \mid \text{V}$$

Every nonterminal from the first production will be present in the LHS in another production. Subsequently, its RHS comprises elements from lower levels of the ontology hierarchy in terms of grammar symbols. In the end, after all tuples are transformed into grammar productions, the CFG is complete. During test generation, concrete values will be mapped to abstract data. For this reason, RHS elements that are connected via is-defined-by will be used as concrete data. For example, by applying the above rules to the nonterminal NN from Table 1, the following production is generated.

$$\begin{aligned}&(\text{NN, is-defined-by, location})\ \text{and}\ (\text{NN, is-defined-by, venue-type})\ \text{and}\\&(\text{NN, is-defined-by, venue-title})\ \text{and}\ (\text{NN, is-defined-by, space})\ \text{and}\\&(\text{NN, is-defined-by, child})\ \text{and}\ (\text{NN, is-defined-by, adult})\ \text{and}\\&(\text{NN, is-defined-by, night})\ \text{and}\ (\text{NN, is-defined-by, star}) \Rightarrow\\&\text{NN} \rightarrow \text{location} \mid \text{venue-type} \mid \text{venue-title} \mid \text{space} \mid \text{child} \mid \text{adult} \mid \text{night} \mid \text{star})\end{aligned}$$

As already mentioned, MRs affect ontologies in a way that its structure remains unchanged. Thus, the CFG remains unaltered for every subsequent ontology. This means

that the CFG is generated only once from the initial ontology. During the next step, i.e. test generation, the generated CFG will be consulted in order to retrieve concrete values for chatbot-intern keywords.

5.2 Test generation & execution

Before the testing process can start, several components are predefined before initiating the process. The source ontology model, ONT_s, contains initial data for test concretization. In addition to that, a structure for sentences, which is understood by the chatbot, is given. This structure, as depicted below, contains the basic linguistic outlook of messages that can be sent by a user. Actually, it contains placeholders in sentences for word classes from the ontology model. These word classes resemble keywords from Section 2, which in turn correspond to data properties in Table 1. For example, the placeholder for a keyword noun is mapped to its counterpart in the instances of the ontology. The same procedure is done with the other word classes. The structure for sentences is given as follows.

```
<sentence>::= I want to <book> a <room> in <place>
<book>::= book
<space>::= room
<place>::= Madrid
<accomodation>::= hotel
<venue>::= Elmhurst
<days>::= today
<adults>::= <number> <guests>
<guests>::= adults
<youngling>::= <number> <children>
<children>::= children
<nights>::= <number> <sleep>
<sleep>::= nights
<rating>::= <number> <points>
<points>::= stars
<number>::= 5
```

After the individual MRs have been specified, the test generation process can begin. The algorithm, **Bot-O-Morph**, as shown in Algorithm 1, represents an extension to the previous algorithm in Bozic and Wotawa (2019). The algorithm adds ontologies to the metamorphic test generation and execution procedures. Test generation starts with the source ontology ONT_s. Then, transformation rules are applied in order to infer the CFG from the ontology model. One sequence of messages from the point of the user is considered to be one test case. It's supposed that the input values for the source test case are denoted as I_s, whereas the follow-up values are given as I_f. Also, their outputs are defined as O_s and O_f, respectively. The obtained output serves as an indicator for the test verdict for each MR. Of course, the output differs for individual inquiries and available information. For example, different numerical inputs will result in different registration information. The type of responses, however, will remain the same. In fact, every test case represents one dialog scenario with the chatbot. It consists of multiple actions, namely a_0-a_7, which contain one or multiple key information. For example, the source test case, I_s, is given in Table 2.

Algorithm 1 Bot-O-Morph – Ontology-based metamorphic test generation and execution algorithm for chatbots

Input: Program P, source ontology ONT_s, set of metamorphic relations $MR = \{\mathtt{mr}_1, \ldots, \mathtt{mr}_n\}$, source test case $I_s = \{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$, a function $\Omega = (MR, ONT_s) \mapsto ONT_f$ that that mutates the ontology, a function $\Psi = (ONT_s) \mapsto G$ that maps the ontology to a grammar G, a function $\Phi = (G, I_s, ONT_f) \mapsto I_f$ that infers follow-up test cases.
Output: Follow-up ontology ONT_f, metamorphic test set $TS = \{\mathtt{I}_{f1}, \ldots, \mathtt{I}_{fi}\}$ where each $I_{fi} = \{\mathtt{a}_0, \ldots, \mathtt{a}_n\}$ and a list with source test cases $V = \{\mathtt{v}_1, \ldots, \mathtt{v}_n\}$.

$TS = \emptyset$
$G = map(\Psi, ONT_s)$
$O_s = exec(I_s, P)$
$V = V \cup \{(I_s, O_s)\}$
for $v \in V$ **do**
  while $MR.hasNext()$ **do**
    $ONT_f = mutate(\Omega, mr_n, ONT_s)$. ▷ Apply MR to ontology
    $I_f = generate(\Phi, G, I_s, ONT_f)$ ▷ Generate test case
    $TS = TS \cup \{I_f\}$
    for $I_f \in TS$ **do**
      $res(I_f) = FAIL$
      $O_f = exec(I_f, P)$ ▷ Execute test case
      if check$((I_s, O_s), (I_f, O_f), mr_n)$ fails **then** ▷ Execute MC
        $res(I_f) = PASS$
        $V = V \cup \{(I_f, O_f)\}$ ▷ Save successful test case
      else
        $res(I_f) = FAIL$
      end if
    end for
  end while
end for

As can be seen, I_s consists of eight actions that are submitted as user requests. Also, every subsequently generated test case will consist of the same number of actions, i.e. contain the same structure. This means that every test case consists of multiple interactions with a chatbot. After every request, the chatbot is programmed to give a corresponding response. During testing, all responses are recorded for every submitted action. On the right side of the table, the chatbot's mandatory parameters are enumerated. The chatbot processes the individual actions and, if possible, assigns provided values to their parameter. At the beginning, the individual actions are manually defined for the source test case I_s. It represents a sequence that ultimately leads to a specific booking reservation. Then, I_s is executed and the chatbot's source output, O_s, is recorded. This represents an important step that substitutes the test oracle: The comparison of O_s to the result from the follow-up test case, O_f, will determine the test verdict. Afterwards, the first MR is applied, thus resulting in the follow-up ontology ONT_f.

In order to generate meaningful tests, a modified version of Grammar Solver Grammar-solver (2018) is used. The implementation applies MRs and traverses through the sentence structure, thereby constructing new user inputs in combination with the CFG from the ontology. As already shown, the CFG is defined by finite sets of terminal and nonterminal symbols. Nonterminal symbols resemble keywords from Section 2, whereas terminals represent concrete values from an ontology. The expression, i.e. a sequence of symbols, guides the generation of an action by assigning concrete values according to a MR. For every MR, instances from the initial ontology are mutated into different values. In turn, the CFG provides building blocks for such test generation. In fact, this process can be subtracted as a mutation process. MRs proscribe unique mutations to individual actions, thus generating modified, i.e. follow-up, test cases. So,

Table 3 Possible follow-up test cases for MR1

#	Action
0:	I want to order a place in Madrid
1:	book a hotel
2:	Elmhurst
3:	I want to check-in today.
4:	1 people
5:	2 kids
6:	1 days
7:	i want 5 ratings

the final shape of a follow-up test cases is determined by the mutation operation and the source test case. For example, Table 3 emphasizes all keywords from I_s that can be separately affected by MR1.

Subsequently, every I_f will completely resemble I_s, with the exception of one single added change in some of its a. In this case, one keyword is exchanged with a synonym for every single I_f. On the other hand, the follow-up test case I_f in Table 4 depicts the case where MR4 is applied to I_s. The intention is to check how minimal changes affect the overall testing result. In such way, a diverse test case is obtained by applying mutations to just one initial test. By doing so, the intention is to obtain tests that are able to lead to states during execution that are not covered by the implementation or the specification.

Follow-up test cases are executed against the SUT and the output is recorded. From the technical point of view, obtained values from Grammar Solver are assigned to HTTP requests and submitted to the chatbot system on the fly. In return, its HTTP responses are processed and handled accordingly. The MCs are done after the final O_f of a test case is received. If no vulnerability is triggered, i.e. if the MC fails, then the test case is added to the list of successful tests V. Otherwise, in case that an issue has been detected, the test will be disregarded. The process continues until all MRs have been applied to ONT_s and checked. In theory, the same process is restarted but with new source tests from the successful ones. However, in this work follow-up test cases are generated just from the initial O_s.0

Table 4 Possible follow-up test cases for MR4

#	Action
0:	I want to exchange a pod in nowhere
1:	book a couch
2:	Shuttle
3:	I want to check-in thousand.
4:	1 alien
5:	2 kobolds
6:	1 insomnia
7:	i want 5 coins

In general, the process can continue until every symbol has occurred for every combination of values for all MRs. It remains the task of the tester to define the termination condition.

6 Evaluation

The proposed approach for functional testing is evaluated on a SUT that processes inputs in natural language. First, a use case demonstrates one test scenario and discusses its output. Then, a broad discussion is given on the obtained test results.

6.1 Use case

A user interaction with the booking chatbot represents an interaction that ultimately leads to a hotel reservation. The communication between both peers is initiated by sending some initial inquiry. Since the chatbot requires different information in order to complete a reservation, it responds with a request to obtain such information. As already mentioned in Section 2, these requests are encoded in form of unique keywords. The chatbot will insist on answering particular questions and should ignore inappropriate values. Usually, the chatbot expects to receive one information for every request. However, the user can submit multiple booking details as part of one request. After the SUT obtains all necessary information, the interaction terminates by summarizing all input data in form of a reservation. In an ideal case, the reservation resembles the provided user input values. Such a scenario is given in Section 2, which shows the interaction that is obtained with the source test case I_s from Table 2. However, the generated test cases from the ontology-based MT approach are meant to confront the SUT with different types of inputs. It remains the task of the chatbot to behave correctly even in unexpected scenarios.

However, the conducted experiments have confirmed that the chatbot does not always behave in an expected manner. For example, the chatbot does in some cases how it should not. Let's demonstrate such an example in a use case with a test case that refers to Table 4. This MR instructs to replace keywords with unrelated terms. In this example, the keyword in a_3 is instantiated with the unrelated term "thousand". Thus, the subsequent interaction with the chatbot looks as follows.

```
I want to book a room in Madrid.
- Do you want a hotel, hostel or apartment?
book a hotel
- Which one do you prefer?
Elmhurst
- When do you want to start your stay?
I want to check-in thousand.
- How many adults?
2 adults
- How many children?
2 children
- How many nights?
1 night
- How many stars do you prefer?
i want 5 stars
- Let me sum up! Location: Madrid, Name: Elmhurst, Check-In:
  1000-01-01, Accommodation: hotel, People: 2 adults, 2 children,
  Nights: 1, Stars: 5 stars
```

As can be seen, the chatbot confirms a reservation, which resembles all user input data. However, the obtained result violates the output relation of MR4, which states that $O_s \neq O_f$. Although the chatbot requests information about the check-in date, it failed to recognize the invalid input value. In such way, it confirmed an invalid reservation as well. The correct behaviour of the chatbot would be to reject the submitted input in the first place. In this concrete case, it can be concluded that the chatbot lacks proper input validation for time data.

6.2 Discussion of results

In total, 104 test cases were generated and submitted against the chatbot. In general, the MRs have a huge impact on the total number of generated test cases. Since the individual actions from I_s differ from each other, the applied MRs cause different results with regard to number and shape of test cases. Each MR is applied individually on just one keyword in an action, so the new test case differs only in one change from its original (the exception being MR5). In MR1, for example, the number of synonyms that are applied to every keyword is kept short. In fact, three synonyms were used on average for every affected keyword. The synonyms that are used for the keyword "guests" in a_4 include "people", "humans" and "adults". This means that three test cases are generated from I_s that focus on this keyword's influence on the test results. Also, no successive source test cases were used, i.e. subsequently using a successful I_f as I_s. Regarding MR2, only natural numbers are taken into consideration. This can be augmented, for example, by introducing rational and decimal numbers. Similar conclusions can be drawn about the influence of other MRs on the number of test cases. Of course, the definition of additional MRs would increase the number of follow-up test cases as well. In MT, the implementation of MRs and termination mechanisms for test generation remains a task of the tester.

Table 5 depicts all results that have been achieved. It depicts the number of test cases for every MR (#total). The number of successful tests (#pass) denotes cases where no fault is

Table 5 Test results

MR	#total	#pass	#fail
MR1	60	56	4
MR2	16	16	0
MR3	10	5	5
MR4	10	5	5
MR5	8	8	0

detected. On the other hand, the number of failing test cases (#fail) shows the occurrence of detected faults per MR.

Since a SUT behaves different for each input, the test results are analysed with regard to every MR.

MR1: Several synonyms were added for keywords, whereas a few discrepancies were encountered with Dialogflow. The initial booking request action was successful in all cases. Thus, the chatbot was able to recognize the intent and finalize the reservation even when the client reformulates the request. A failing test case was encountered with a_{4-7}, where synonyms like "guests" instead of "adults", "kids" instead of "children" and "ratings" could not be recognized by the chatbot.

MR2: This MR is unique among the relations because it affects only actions with numerical values. That is, only four out of eight inputs are addressed by the mutation. It can be concluded that changing the number in an action does not cause any failures. The chatbot succeeds with generating a reservation with different values. Therefore, all tests have been successful.

MR3: Omitting words causes a lot of confusion in the chatbot. Initiating a reservation with a_0 is possible even without the explicit mentioning of "book" or "room". However, if no city name is provided, the client cannot proceed further. The conclusion is that the explicit statement of a city name at the beginning of the communication is mandatory. Also, if no accommodation type, check-in dates and stars are provided, then the reservation fails. The chatbot keeps insisting for the information and ignores further user requests. For actions with numerals, numbers were kept but words to their right were omitted. Interestingly, a valid information is made when providing just numerals without "adults", "children" or "nights". It seems that the chatbot assumes that a client always provides information in a strict order. Thus, it assigns the information to parameters without further clarification. However, this seems not to be the case when "stars" is omitted.

MR4: Replacing keywords with nonrelated terms triggered strange behaviour on side of the chatbot. For example, formulating a_0 into "I want to *exchange* a *pod* in *nowhere*" (three follow-up test cases, one for every keyword) resulted in a valid registration! If behaviour is compared to MR3, the chatbot seems to accept every value for city name as a valid one. This is an important implementation flaw in the chatbot. Actions a_{2-4} are processed as expected by the SUT. On the other hand, "alien" and "kobolds" seem to be a valid substitute for "adults" and "children" in a_4 and a_5, probably due to the reasons mentioned in MR3. However, a_6 and a_7 have been rejected, as demanded by the relation.

MR5: This MR is unique since it requests a reordering of actions in a test case. This means that changes are just added to the abstract testing layer. Thus, ONT_s remains unaffected by this MR, so that concrete values will not differ between I_s and I_f. A randomized approach was adapted that resulted in eight test cases with different sequences

of actions. As expected, every O_f from SUT differs from O_s. The chatbot's response depends on the first user inquiry. Usually, the SUT keeps asking for a certain information and remains in a specific queue. If the requested information is received in the meanwhile, then it switches to another queue. Also, if the chatbot encountered unexpected input (from its point of view), then unexpected responses were sent. This included an empty text or unrelated sentences (e.g. "Talk to you soon!").

Since MT is applied for functional testing purposes, the intention was not to exploit a SUT. Due to the reason that a tester does not know about the inner workings of a SUT and its available information, some guessing is needed. The tester can assume that some keywords will be understood by the chatbot, others can be completely avoided. In this approach, unintended behaviour indicates that 1) an implementation flaw or oversight or 2) insufficient information on side of the Dialogflow implementation. Actually, here both observations affirm known advantages of MT, namely the detection of verification and validation defects, respectively Chen et al. (2018). In the first case, it can be assumed that the chatbot did behave to its implementation or incomplete specification. However, it failed to cover cases, which it should be able to understand. Also, the assignment of information to a specific parameter without clear indication can be considered a drawback as well. Finalizing a reservation with unsuitable data (e.g. fictional city names or substitutes) is even worse.

The test results demonstrated that the chatbot concluded reservations even with missing information in a nonintuitive manner. This means that the chatbot does something it should not. The same observation was proven when testing against MR3. On the other hand, MT detected that the SUT does not what it should do (e.g. with MR1). Also, meaningless input is wrongly associated with an intent with MR4. The assumption is that the reason why unrelated terms are still "understood" due to the fact that the chatbot follows its strict order and ignores other information ("stars" being an exception). The recognition of terms depends upon the pre-specified entries in the chatbot's entity database. A different set of values and intents would likely result in different behaviour of the system. Interestingly, some natural language input does disturb the chatbot as well. Adding a "I want to" to mandatory information does not match the chatbot's intent. It seems that the SUT follows a minimalistic approach.

The obtained results indicate that the metamorphic approach is able to detect unexpected behaviour in a system. Also, this work confirms the known advantage of ontologies, i.e. their reusability in different scenarios. Also, it uncovers clues about inner workings of the chatbot and provides some clues about the encountered issues. Although the chatbot did succeed to ensure functionality in some cases, other test cases triggered situations, which the SUT failed to handle correctly. Consequently, the reasons for these issues must be addressed separately.

7 Related work

Research that correlates with the presented work includes ontologies, metamorphic testing and chatbot testing. Ontologies are used in several domains but this section focuses on ontology-based test generation for NLP systems. Works that focus on the second topic are often conducted on industrial and online applications. Also, these approaches often interact

with external implementations and resources. On the other hand, the literature that focuses on testing of chatbots is sparse.

7.1 Ontologies in NLP

Early examples of domain-specific ontologies for formalization in NLP include ONTOS Carlson and Nirenburg (1990) and SENSUS Knight et al. (1995). On the other hand, early works that describe domain-independent and reusable models count Penman Upper Model Bateman et al. (1990) and Ontolingua Gruber (1992). The former of the application independent approaches describes an upper ontology hierarchy that comprises general semantic categories. In this case, classification functions associate optional domain-specific knowledge with key elements from the upper model. The latter language, however, enables that user can access and manipulate new ontologies from existing libraries. Also, the known long-term Cyc project Lenat et al. (1985) implements a knowledge base in form of an ontology and remains still in use today Cycorp (2020).

The authors of Al-Zubaide (2011) implement their idea, where NLP ontologies are transformed into knowledge bases for chatbots. The initial OWL ontology is transformed into a relational database by following a set of mapping rules. The proposed domain-independent chatbot, OntBot, processes user inputs and checks for matches in this underlying database. Similar to our implementation, the ontology is used in order to match specific keywords for further processing.

The work in Cimiano et al. (2013) have implemented a Natural Language Generation (NLG) system that relies on ontologies. Raw RDF data are converted into natural language text by addressing an ontology lexicon that encompasses context-specific lexical information. In addition to that, a domain corpus encompasses indexed structure with expert information in form of parse trees and leafs. The retrieved information from this structure is weighted, thus obtaining statistical information from the domain. The ontology contains rule-based expressions that will be used for verbalization. Natural language is generated by applying microplanning, i.e. checking against the rules of the ontology lexicon, thereby using statistical information from the previous step. Similar to the presented paper's approach, the ontology is feasible for small domains.

Also, Nguyen et al. (2019) introduced two NLP systems that produces natural language from ontologies. The systems rely on grammars and existing data sources in order to generate valid sentences. Ontologies are defined by relying on a vocabulary that is implemented in Prolog. The NLP systems extract information like nouns, adjectives and verbs from the ontology. Subsequently, a parser analyses the produced output and sentences are constructed according to the structure that is defined by a grammar. Similar to this work, ontologies are used in order to extract information and generate new inputs. However, the input generation differs from this paper's approach and does not represent a testing approach.

An ontology-based chatbot that helps a customer support service was introduced in Silva et al. (2019). A Protégé ontology is used for semantic search and suggests automated answers for the client. The use of such a system reduces the overall response time without affecting the quality of the service.

The authors of Jung and Kim (2020) address a Question Answering (QA) system based on ontologies in order to generate queries from natural language. First, a user input is segregated into several tokens that catch specific keywords. Then, these tokens are mapped to resources from an ontology. Subsequently, a search algorithm generates

query graphs and identifies relationships between resources. In turn, these graphs are relied on in order to produce SPARQL queries for semantic search.

Philosophical discussions about ontologies can be found in Moltmann (2017) and Smith (2014). Whereas the former puts an emphasis on semantics of natural language, the latter discusses ontologies in computer science. Another detailed analysis of NLP and ontologies is given in Cimiano et al. (2014).

7.2 Metamorphic testing

The preliminary work that addressed oracle-free programs is discussed in Weyuker (1982). Here, the notion of *pseudo-oracles* is used, i.e. external programs that substitute a real test oracle. Additional programs are implemented and tests are run against both the original and the substitute program. If the results match, then the results of the original program are considered to be valid.

In Segura et al. (2015) the authors elaborate a MT-based approach for testing of *feature models* (FM) and *variability models* (VM). Such models encompass the definition of specific products by means of features or configurations, respectively. MRs are defined between models and their inferred products, and a test generator. New models are generated from previous ones by adding mandatory features. In this way, many models and valid configurations can be generated automatically.

The authors of Chen et al. (2016) address the applicability of MT to cybersecurity. In this approach, MRs are defined in order to test for web failures. Several real-world obfuscators have been tested by using a small test suite. The conducted evaluation indicates that several bugs were found.

Segura et al. (2018) presents a paper that tests Web Application Programming Interfaces (APIs) that rely on the REST architecture. First, the authors define MRs by means of Metamorphic Relation Output Patterns (MROPs). That is, they recognize general patterns in all RESTful Web APIs and apply corresponding MRs that satisfy the patterns. The approach was tested on two known web applications, thereby obtaining positive results.

The work in Pesu et al. (2018) applies a MT approach for machine translation services. It uses a pre-scanned data source in order find correct translations in different languages. In addition to that, it applies comparison metrics to estimate the quality of translations. In such way, the approach is able to identify bad translations so typical human translation flaws can be avoided. Another paper that applies MT to NLP is presented in Ma et al. (2020). A MT approach is defined to test NLP models for discriminatory language inputs. The framework, MT-NLP, automatically identifies problematic noun phrases in sentences. It applies MRs to replace unfair nouns with analogous terms that don't provoke fairness violations. Instead of ontologies, their framework relies on information from existing knowledge graphs. Also, it calculates the distances between words in order to generate new inputs. Another work that addresses MT for machine translators is given in Yan et al. (2019). Here a MR applies translations to different sentences and checks for semantic (dis)similarity between languages. Inputs are generated that are meant to be homologous in different languages. In contrast to the presented work, they validate the produced inputs from the translator and ignore the SUT that processes these inputs.

7.3 Testing of chatbots

In contrast to the plentiful number of MT-related papers, only a few papers address testing of chatbots. For example, Vasconcelos et al. (2017) that deals with testing of conversational systems with regard to functionality. A testing system, called Bottester, emulates users that interact with a chatbot. A large amount of pre-defined user inputs is submitted to the chatbot in an automated manner. Bottester also applies invariants to user inputs, like synonyms and typos, and executes them automatically. In contrast to the presented work, it puts more focus on performance than detection of functional incorrectness in a chatbot.

In the previous work in Bozic (2019), an approach for functional testing of a hotel booking chatbot was introduced. AI planning is applied for generation of test cases and the chatbot's output is compared to an expected result in an automated manner. However, this work extends the approach by adding a greater input generation foundation.

Techniques that resemble the test generation technique from this paper are given in Ruane et al. (2018) and Guichard et al. (2019). The idea behind the proposed technique is to change valid user inputs by paraphrasing them. Different techniques are used for this sake in order to retrieve a divergent input from the original one. The obtained results indicate that the SUT does not always recognize an intent when words are replaced with synonyms. Consequently, it might be claimed that the results from MR1 complement the observations from their approach. On the other hand, the main difference represents the fact that their technique relies on known expected values. This stands in stark contrast to a MT-based approach. In the presented paper, the use of multiple MRs yield more insight into the inner workings of the SUT.

The authors of Bravo-Santos et al. (2020) implement a methodology for generation of inputs for context-dependent conversations. Inputs are defined in a structure that matches the intent and expected responses from a chatbot. Necessary parameter values are generated by applying operations, like fuzzing and mutations, to the training phrases. Similar to the presented paper, the approach targets chatbots that are implemented in Dialogflow. In such way, the authors generate inputs of quality for a structured conversational context.

8 Conclusion and future work

In this paper, an emerging issue in software testing is addressed, namely testing of AI systems. For this reason, a testing approach for chatbots is introduced. The approach applies ontologies to a metamorphic testing technique. In general, ontologies represent conceptualizations of domain knowledge. In such way, they provide information for specific purposes. Since the output of an AI system is difficult to predict, the approach introduces metamorphic checks instead of traditional test oracles. The approach applies MRs to ontologies for metamorphic test generation and execution for functional testing of chatbots. In the aftermath, the approach is evaluated on a chatbot system from the tourism industry.

Since natural language represents a structured entity, ontologies can be modelled in order to resemble this structure. They represent models that provide data and set relations between data classes. By doing so, ontologies produce different semantic interpretations of a sentence. In such way, they provide a foundation for the generation of structured and meaningful inputs. For a NLP system, the ontology-based approach gives an indication

about why an error occurred. In such way, it supplements the MT approach. The strength of MT lies in the fact that it proves an issue in an uncertain environment. However, it often fails to indicate the reason of the error. In such way, ontologies can supplement MT in search for root causes. Also, by applying MT, new ontologies can be inferred. In such way, new input models are inferred from a solid foundation.

Merits and pros:

- In testing, ontologies supplement the test generation process. In such way, it provides input data for concrete test cases. In the aftermath, an ontology provides evidence about what type of information caused issues during execution. For example, let's assume that a synonym could not be handled correctly by the SUT. The issue can be traced back to the source data class in the ontology. In such way, it can be assumed that other synonyms might be problematic as well. This observation is confirmed by the results from the evaluation. The detection of such issues is especially important in NLP.
- Ontologies can be easily extended with new data, thereby not affecting the structure.

Demerits and cons:

- In comparison to the previous paper in Bozic and Wotawa (2019), the introduction of an ontology model did not detect new defects. However, as already mentioned, the ontology helps in finding its root causes in the NLP domain.
- Since ontologies formalize complex systems, some implementation effort and knowledge is needed.

Finally, the promise of the presented approach is that it can be applied to test functional correctness of a chatbot. In the future, ontologies and MT can be used for testing of nonfunctional properties as well. The ontology can be used to conceptualize structured data from other domains than NLP systems. In such way, it can be used as an input model for test generation. On the other hand, MT-based approaches can be applied under circumstances where expected behaviour is difficult or impossible to determine. New MRs can be defined for existing problems, especially with regard to testing of AI systems. Especially for the latter, the combination of ontologies and MT can be used to detect issues in indeterministic environments.

Acknowledgements The research presented in the paper has been funded in part by the Austrian Research Promotion Agency (FFG). I want to thank the anonymous reviewers for their constructive feedback, which was addressed in the paper. In addition to that, I would like to express my gratitude to this journal's special issue guest editors for their kind approach.

Funding Open Access funding provided by Graz University of Technology

References

Cycorp Inc. https://www.cyc.com/. Accessed 29 Jan 2020

Dialogflow. https://dialogflow.com/. Accessed 11 Dec 2018

Grammar-solver. https://github.com/bd21/Grammar-Solver. Accessed 13 Aug 2018

Oxford Reference. https://www.oxfordreference.com. Accessed 03 Feb 2020

RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/. Accessed 17 Feb 2020

Web Ontology Language (OWL). https://www.w3.org/OWL/. Accessed 05 Dec 2019

WordNet - A Lexical Database for English. https://wordnet.princeton.edu. Accessed 04 Feb 2020

Al-Zubaide, H., & Issa, A. A. (2011). OntBot : Ontology based ChatBot. In *Proceedings of the Fourth International Symposium on Innovation in Information & Communication Technology*

Bateman, J. A., Kasper, R. T., Moore, J. D., & Whitney, R. A. (1990). *A General Organization of Knowledge for Natural Language Processing: the Penman Upper Model*. USC/Information Sciences Institute, Marina del Rey, California: In Technical report.

Bozic, J., Tazl, O. A., & Wotawa, F. (2019). Chatbot Testing Using AI Planning.In Proceedings of the International Conference on Artificial Intelligence Testing (AITest)

Bozic, J., & Wotawa, F. (2019). Testing Chatbots using Metamorphic Relations. In *Proceedings of the 31th IFIP WG 6.1 International Conference on Testing Software and Systems (ICTSS'19)*

Brandtzæg, P. B., & Følstad, A. (2017). Why People Use Chatbots. In *Proceedings of the 4th International Conference on Internet Science (INSCI'17)*

Bravo-Santos, S., Guerra, E., & de Lara, J. (2020). Testing chatbots with Charm. In *Proceedings of the 13th International Conference on the Quality of Information and Communications Technology (QUATIC'20)*

Carlson, L., & Nirenburg, S. (1990). World modeling for NLP. In *Technical Report CMU-CMT-90-121, Center for Machine Translation, Carnegie Mellon University, Pittsburgh, PA*

Chen, T. Y., Cheung, S. C., & Yiu, S. M. (1998). Metamorphic Testing: A New Approach for Generating Next Test Cases. In *Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong*

Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T. H., & Zhou, Z. Q. (2018). Metamorphic Testing: A Review of Challenges and Opportunities. In *ACM Computing Surveys (CSUR), Vol. 51, Issue 1*

Chen, T. Y., Kuo, F.-C., Ma, W., Susilo, W., Towey, D., Voas, J., & Zhou, Z. Q. (2016). Metamorphic Testing for Cybersecurity. In *Computer, Vol. 49 , Issue 6*

Cimiano, P., Lüker, J., Nagel, D., & Unger, C. (2013). Exploiting ontology lexica for generating natural language texts from RDF data. In *Proceedings of the 14th European Workshop on Natural Language Generation*

Cimiano, P., Unger, C., & McCrae, J. (2014). *Ontology-Based Interpretation of Natural Language*. In Synthesis Lectures on Human Language Technologies: Morgan & Claypool Publishers.

Følstad, A., & Brandtzæg, P. B. (2017). Chatbots and the New World of HCI. *interactions*, 24(4), 38–42

Genesereth, M. R., & Nilsson, N. J. (1987). Logical Foundations of Artificial Intelligence. In *Morgan Kaufmann*

Gómez-Pérez, A. (1999). Ontological engineering: A state of the art. *Expert Update,* 2(3)

Gruber, T. R. (1992). Ontolingua: A Mechanism to Support Portable Ontologies. In *Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory*

Gruber, T. R. (1993). A translation approach to portable ontology specifications. In *Knowledge Acquisition, Vol. 5, Issue 2*

Grudin, J., & Jacques, J. (2019). Chatbots, Humbots, and the Quest for Artificial General Intelligence. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'19)*

Guarino, N., & Musen, M. A. (2005). Applied ontology: Focusing on content. *Applied Ontology,* 1(1)

Guichard, J., Ruane, E., Smith, R., Bean, D., & Ventresque, A. (2019). *Assessing the Robustness of Conversational Agents using Paraphrases*. In IEEE: University College Dublin.

Jung, H., & Kim, W. (2020). Automated conversion from natural language query to SPARQL query. In *Journal of Intelligent Information Systems*

Knight, K., Chander, I., Haines, M., Hatzivassiloglou, V., Hovy, E., Iida, M., Luk, S. K., Whitney, R., & Yamada, K. (1995). Filling Knowledge Gaps in a Broad-Coverage Machine Translation System. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*

Lenat, D., Prakash, M., & Shepherd, M. (1985). CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *AI Magazine,* 6(4)

Ma, P., Wang, S., & Liu, J. (2020). Metamorphic Testing and Certified Mitigation of Fairness Violations in NLP Models. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*

Mauldin, M. L. (1994). ChatterBots, TinyMuds and the Turing Test: Entering the Loebner Prize Competition. In *AAAI '94 Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 16–21

Moltmann, F. (2017). Natural Language Ontology. In *Oxford Encyclopedia of Linguistics*

Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. In *AI Matters, Vol. 1, Issue 4*

Nguyen, V., Son, T. C., & Pontelli, E. (2019). Natural Language Generation from Ontologies. In *Proceedings of the International Symposium on Practical Aspects of Declarative Languages (PADL'19)*

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. In *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*

Pereira, M. J. V., Fonseca, J., & Henriques, P. R. (2016). Ontological approach for DSL development. In *Computer Languages, Systems & Structures, Vol. 45*

Pesu, D., Zhou, Z. Q., Zhen, J., & Towey, D. (2018). A Monte Carlo Method for Metamorphic Testing of Machine Translation Services. In *Proceedings of the 2018 ACM/IEEE International Workshop on Metamorphic Testing (MET)*

Ruane, E., Faure, T., Smith, R., Bean, D., Carson-Berndsen, J., & Ventresque, A. (2018). BoTest: a Framework to Test the Quality of Conversational Agents Using Divergent Input Examples. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion (IUI'18 Companion)*

Segura, S., Durán, A., Sánchez, A. B., Le Berre, D., Lonca, E., & Ruiz-Cortés, A. (2015). Automated metamorphic testing of variability analysis tools. In *Software Testing, Verification and Reliability, Vol. 25, Issue 2*

Segura, S., Parejo, J. A., Troya, J., & Ruiz-Cortés, A. (2018). Metamorphic Testing of RESTful Web APIs. In *IEEE Transactions on Software Engineering, Vol. 44, Issue 11*

Segura, S., Fraser, G., Sánchez, A. B., & Ruiz-Cortés, A. (2016). A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering, 42,* 9

Silva, M. F. B., Yaguinuma, C. A., dos Santos, F. J. J., & Boalim, T. (2019). Desenvolvimento de um Chatbot baseado em Ontologia para Atendimento a Chamados de Suporte ao Cliente. *Revista Eletrônica de Iniciação Científica em Computação,* 17(3)

Smith, B. (2014). The Relevance of Philosophical Ontology to Information and Computer Science. In *Ruth Hagengruber & Uwe Riss (eds.), Philosophy, Computing and Information Science. Chatto & Pickering*

Turing, A. M. (1950). Computing Machinery and Intelligence. In *Mind, New Series, Vol. 59, No. 236 (Oct., 1950)*, pages 433–460

Vasconcelos, M., Candello, H., Pinhanez, C., & dos Santos, T. (2017). Bottester: Testing Conversational Systems with Simulated Users. In *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems (IHC 2017)*

Weyuker, E. (1982). On Testing Non-Testable Programs. In *The Computer Journal, Vol. 25, No. 4*

Yan, B., Yecies, B., & Zhou, Z. Q. (2019). Metamorphic Relations for Data Validation: A Case Study of Translated Text Messages. In *Proceedings of the 4th International Workshop on Metamorphic Testing (MET'19)*

Josip Božić worked as a postdoctoral researcher at Graz University of Technology, Institute of Software Technology, during the writing of this paper. His research interests include cybersecurity, artificial intelligence and model-based testing of web applications, network protocols and NLP systems.