Towards an Ontology for UML State Machines

Yousef Belgueliel, Mustapha Bourahla, and Mourad Brik

Abstract—Ontology is a conceptual model that is used to represent the concepts in a domain and relationship between the concepts. It can be used for sharing and reuse of knowledge that allows humans and machines to exchange diverse information. UML state machines used to describe the behavior of a software systems. This article aims at to provide a solution for representing UML state machine model as an ontology expressed in OWL. a method proposed is a conceptualization of UML state machine and its operation, in order to check its consistency and its conformity, we chose OWL to represent formally our state machine augmented with SWRL rules to represent the dynamic aspect of the operation system and SPARQL to query our ontology.

Index Terms—State machine, semantics checking, OWL DL, SWRL.

I. INTRODUCTION

UML which is a modeling language recommended by the OMG provides several views to software designers for modeling and analyzing the different aspects of software system[1]. And UML is the important part of MDSD (Model-Driven Software Development) which is a popular method of software development [2]. Based on the above characteristic, UML is widely used in the software development. Especially, the UML state machines is mainly used to express the behaviors of dynamic systems [3].

Ontology is a conceptual model which used to represent the concepts in a domain and relationship among them [4]. Researchers propose the use of ontologies for sharing and reuse of the knowledge that allows humans and machines to exchange diverse information [5]. Owl (web ontology language) [6] is a current standard for specifying such ontologies. The UML state machine models that contain information, which can be stored using a knowledge representation format like ontology. Currently, a limited research is available on how we should represent the behavioral knowledge captured in an UML state machine model using the OWL ontology language.

Our present objective is to propose an ontology for UML state machine model, that will be instantiated, provides the ability to reason and to do queries. The paper is organized as follows. Section II introduces the preliminaries concepts underlying state machine diagrams, ontology, OWL (Web Ontology Language) and SWRL (Semantic Web Rule Language). The conception of ontology and the reasoning on this ontology is defined in Section III. and finally we conclude and define some future works.

II. PRELIMINARIES

A. UML State Machine Diagrams

UML behavioral state machine consists of state and transition, and used to describe the behavior of dynamic systems [3]. They are a variant of Harel's statecharts [7]. We analyze three kinds of basic constructs in UML behavioral state machine, i.e. state, transition and pseudostate.

1) States

States in a UML behavior state machine have three types, i.e. simple state, composite state and submachine state. Composite state is further divided into orthogonal composite state and composite state depending on whether it has exactly one or more than one regions. A submachine state is semantically equivalent to a composite state [3]. Currently we support simple state and composite state (including orthogonal composite state). In our paper, we plan to support submachine state since it can be converted to a composite state.

Final state is a special kind of state. It does not have regions, entry/exit behaviors or do activities. But it still belongs to the class of state, since it has a fundamental difference with pseudostate, viz an object can temporally stay in a final state while it cannot stay in any of the pseudostate.

2) Transitions

Transitions are viewed in UML as relationships between two states indicating that an object in the first state will enter the second state and perform specifications when a specified event occurs provided that certain conditions as satisfied [3]. A transition in a state machine may be labeled by a string of the form e[c]/a, which means that the occurrence of event e, when the guard condition c is true, triggers the firing of the transition, as a result of which the object performs sequence of actions a. UML state machine include internal, external and local transitions. an transition can emanate and target a pseudostate as well, and an compound transition can be composed of a series of transitions with more than one pseudostates in between the source and target states. In general, a compound transition is an acyclical unbroken chain of transitions joined via join, junction, choice or fork pseudostates. We illustrate a compound transition in Fig. 1.

As is shown in the Fig. 1, we can have a compound transition connected by a junction an choice pseudostates. Actually, according to the definition of a compound transition, we can have a compound transition connected by any number of junction, choice and join pseudostates with

Manuscript received July 7, 2013; revised September 16, 2013.

The authors are with Department of Computer Science, Faculty of Mathematics and Informatics, University of M’sila, Algeria (e-mail: bel_you@yahoo.fr, mbourahla@hotmail.com, mourad50200@yahoo.fr).
one fork pseudostate (since fork pseudostate can only target states in orthogonal regions).

Fig. 1. Multiple pseudostates in a compound transition.

3) **Pseudostate**

Pseudostates are introduced to connect transitions to form compound transitions which have a more complete semantic meaning, or to aid the construction of a state machine to improve the expressiveness of it. There are 10 kinds of pseudostates specified in [3] (initial, entryPoint, exitpoint, deephistory, shallowHistory, fork, join, junction, choice, terminate).

**B. Ontology**

The earliest idea of ontology derives from Aristotle where Metaphysics is the knowledge of exploring the being (http://en.wikipedia.org/wiki/Ontology/). It mainly describes the existence of instances or things in the real world. In recent years, ontology is often used in the field of Computer Science and Artificial Intelligence [8], [9], [10] defined ontology as the “An ontology is an explicit specification of a conceptualization”. Ontology is expressed by the aggregation of conceptualizations and a systematic description. Choi et al. classify ontology as three types in ontology: Global Ontology, Local Ontology, and Domain Ontology [11]-[13]. In this paper, we will construct a domain ontology. Ontology is used to clearly describe the concept in a field, the characteristics of properties, attributes, as well as specific restrictions relevant to the concept described. Ontology includes concepts, relationships and instances listed in the following:

1) Concept or Class: A series of concepts that represent topics or characters in the domain ontology.
2) Relationship or Attribute: Relations between concepts when considering a specific concept.
3) Instance: A series of concepts and relationships that have specific knowledge, such as web pages, documents and so on.

For example, ontological elements included class, attribute, and instance in this study. Class refers to a category or concept, such as state, transition, event, and action. Each one can be called a class. Attributes in the ontology are used to describe the relationship between concepts. Instances in ontology are a case of concepts or categories. The instance will inherit all the attributes or relationship of their class.

**C. Web Ontology Language (OWL)**

OWL is the framework proposed by W3C [14]. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDFS) by providing additional vocabulary along with a formal semantics [15]. For example, rdfs:subClassOf in OWL can still be used. RDF lacks the description of relationships, which can be complemented by OWL. OWL can be divided into three levels of language: OWL Full, OWL DL, and OWL Lite.

1) OWL Full: is the maximum expressiveness and the syntactic freedom. However, the RDF base has no computational guarantees in OWL Full. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full (http://www.w3.org/TR/owl-guide/).

2) OWL DL (Description Logic): is a sub-category of OWL language. OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time) (http://www.w3.org/TR/owl-guide/).

3) OWL Lite: OWL Lite provides a quick migration path for thesauri and other taxonomies. It only permits cardinality values of 0 or 1 (http://www.w3.org/TR/owl-guide/) and OWL Lite supports those users primarily needing a classification hierarchy and simple constraints.

**D. Semantic Web Rule Language (SWRL)**

![Fig. 2. An example of SWRL.](image)

![Fig. 3. Class diagram of ontology.](image)
The SWRL was used to describe the relationship between the rules. SWRL is based on semantic rules [16]. Rules of the SWRL are evolved from Rule Markup Language (RuleML) [17]. SWRL rule has an antecedent part and a consequent part to reason where the results combined with OWL ontology. SWRL is a specification of W3C at present time. The basic pattern, which used instance to express the inferential results and concept and relationship the inferential premise in RuleML, was also retained in SWRL (http://www.w3.org/Submission/SWRL/). SWRL may be regarded as the combination of rules and ontology, through which the relationships and terms described in ontology can be used directly when writing rules. At first, the relationship between these classes would have additional rules to be described, but the description in ontology may be used directly in SWRL [18]. There is a relationship of cause and effect between them. Fig. 2 indicates that if x has parent y, x described, but the description in ontology may be used directly when writing rules. At first, the relationship between these classes would have additional rules to be described, but the description in ontology may be used directly in SWRL [18]. There is a relationship of cause and effect between them. Fig. 2 indicates that if x has parent y, x has brother z, based on SWRL to build the rule, then z is y’s uncle.

III. CONCEPTION OF THE ONTOLOGY

An Ontology is specification of the conceptualization and corresponding vocabulary used to describe a domain [10]. It represents a domain knowledge in an understandable way for both human and computer. It is formed by a set of concepts which are organized hierarchically and defined by properties. The conceptual modeling of the UML state machine model is less likely to change than the needs of a user. Thus, the ontology structure (concepts and properties) can be defined and fixed by human study of the domain knowledge of UML state machine model. Contrariwise the ontology instantiation will vary depending on both state machine configuration and requirement descriptions.

In the following we will focus on defining concepts and properties we have chosen.

A. Concepts

From the UML state machine diagrams section, it appears that we need to define concepts that represent State, Transition, Simple, Composite, region, Initial, Final, Action, Condition, Event, Pseudostate, Join, Fork, junction, History. Fig. 3 shows the class diagram of ontology.

B. Relationships between Concepts

Relationships (also known as relations) between concepts in an ontology specify how concepts are related to other concepts (Fig. 3). The set of relations are: Source, Target, Exit, Entry, Do, Effect, Guard, Trigger.

C. Axiom

An axiom is a sentence in first order logic (F-logic) that is assumed to be true without proof. In practice, axioms can be used to refer to the sentences that cannot be represented using only values of a concept. The following sentences represent an extract of axioms for our ontology.

1) A simple state is a state without any regions.
2) A composite state is a state with at least one region.
3) An orthogonal state is a composite state with at least two regions.
4) Only composite state can have entry or exit pseudostates defined.
5) A fork segment must not have guards or triggers.
6) A join segment must not have guards or triggers.
7) A fork segment must always target a state.
8) A join segment must always originate from a state.
9) Transition outgoing pseudostates may not have a trigger (except for those coming of initial pseudostate).
10) All transitions incoming a join segment must originate in different regions of an orthogonal state.
11) A transition is a directed relationship between a source state and target state, it may be part of a compound transition.
12) Pseudostates are typically used to connect multiple transition to form compound transitions.

D. Reasoning on the Ontology

During OWL reasoning, inferences are made, classifying instances of the ontology and associating new properties to instances while maintaining logical consistency.

1) Classification and Assertion: OWL axioms are used for the hierarchical organization of concepts and for the classification of individuals. Thus assume C1, C2 are concepts and P1, P2 are properties. OWL defines two kinds of axioms.

• inclusion axiom C1 ⊆ C2 (P1 ⊆ P2), e.g. Composite ⊆ State, each element of C1 is an element of C2 i.e. C1 is a subclass of C2.
• Equality axiom C1 ≡ C2 (P1 ≡ P2) e.g. Composite ≡ State ∩ ¬Simple, each element of C1 is an element of C2 and vice versa.

To write more expressive conditional rules, we used the Semantic Web Rule Language (SWRL). In our studies, the dynamic semantics of a state machine is captured by the execution of RTC (Run To Completion) steps, which have two kinds of effects, viz, changing active states and executing behaviors. For example, if a transition is enabled and is selected to fire, the following steps are carried out in order:

i) the source state is left. ii) the Actions are executed. iii) the target state is entered. These rules are described by SWRL rule as follow:

Rule 1: Source(?s,?t), Event(?t), Guard(?t)⇒ Exit(?s), it means that the transition t is enabled and its trigger is offered and, if the Guard t is satisfied, then the source state s is exited.

Rule 2: Event(?t), Guard(?t), Target(?t,?d)⇒ Entry(?d), it means that the transition t is enabled and its trigger is offered and, if the Guard t is satisfied, then the target state d is entered.

Finally, we construct an ontology using OWL DL and SWRL, and the ontology contains the semantics of state machine models.

2) Consistency checking: according to the class diagram of ontology (Fig. 3), there are multiplicity constraints for the source end and target end of every relationship. If the state machine model disobeys these constraints, then the state machine model is inconsistent. These constraints are formalized to the multiplicity for domain and range of relationship in the ontology. If the state machine model is inconsistent, then the ontology containing the semantics of state machine model is also inconsistent.
This kind of inconsistency in ontology can be checked by reasoning of ontology according to the following rules:

- A transition connects two states (Source state and target state).
- A transition can include a triggering event, a guard and actions to be executed.
- A fork segment should not have guards or triggers.
- A join segment should not have guards or triggers.
- A fork segment should always target a state.
- A join segment should always originate from a state.
- Transitions outgoing pseudostates may not have a trigger.

The reasoning of ontology according to the above rules can be implemented by Pellet which is a reasoning tool based on the algorithm-Tableaux.

3) **Conformity checking:** to verify conformity of the state machine, we query our ontology looking for wrong or missing specifications. We do this using SPARQL for querying OWL ontologies, providing SQL-Like operations to retrieve knowledge from OWL. Thus, we can check whether a model may not implement some function (e.g. an state cannot be simple and composite at the same time). The reasoning tool-Pellet can query ontology using the querying rules and then return the result to the model designer.

The ontology building is realized under Protége (Fig. 4) and the consistency and conformity checking are implemented under a java application using Jena, a java framework for building semantic web applications.

**REFERENCES**


Youcef Belguediel was born at Djorf, M’sila, Algeria in 1972. He got an engineer degree in information systems in 1998 from INI (National School Computer Science), Algiers, Algeria; a magister degree in Communication and information systems in 2010 from University of M’sila, Algeria. Currently, He is an assistant Professor with the department of computer science at the University of
M'sila, Algeria. His areas of interests include Ontology, artificial intelligence, semantics web and Complex systems.

Mustapha Bourahla has Habilitation from University of Annaba, Algeria (2010), he has PhD degree in computer science from University of Biskra, Algeria (2007) and he has the Master degree in computer science from University of Montreal, Canada (1989). He was a member of the VHDL group at Bell-Northern Research, Ottawa, Canada (1989-1993). He worked for Bell Canada for one year. He was teacher-researcher at University of Biskra (Algeria), from 1994 until 2009. Now, he is teacher-researcher at University of M'sila (Algeria). He has publications in the domains of VLSI and formal methods. His current research interests are formal methods, especially model checking critical systems, semantics web and business intelligence. Dr. Bourahla is a member of a research group working in the domains of semantics web, decision support systems and formal methods at the University of M’sila (Algeria).

Mourad brik is a full assistant teacher of computer science at the University of M’sila he is graduated from the Computer Science Department, Science Faculty, University of Setif, Algeria. He is currently working toward the Doctorat degree in the field of semantic technologies at the University of Setif. His interests range in informatics applications with emphasis on information extraction, knowledge representation, and human-computer interaction and computer assisted learning.