From First-Order Logic to Assertional Logic

Yi Zhou[1,2(✉)]

[1] School of Computing, Engineering and Mathematics,
Western Sydney University, Sydney, Australia
y.zhou@westernsydney.edu.au
[2] School of Computing Science and Technology, Tianjin University, Tianjin, China

Abstract. First-Order Logic (FOL) is widely regarded as the foundation of knowledge representation. Nevertheless, in this paper, we argue that FOL has several critical issues for this purpose. Instead, we propose an alternative called assertional logic, in which all syntactic objects are categorized as set theoretic constructs including individuals, concepts and operators, and all kinds of knowledge are formalized by equality assertions. We first present a primitive form of assertional logic that uses minimal assumed knowledge and constructs. Then, we show how to extend it by definitions, which are special kinds of knowledge, i.e., assertions. We argue that assertional logic, although simpler, is more expressive and extensible than FOL. As a case study, we show how assertional logic can be used to unify logic and probability.

1 Introduction

Classical First-Order Logic (FOL) is widely regarded as the foundation of symbolic AI. FOL plays a central role in the field of Knowledge Representation and Reasoning (KR). Many of its fragments (such as propositional logic, modal and epistemic logic, description logics), extensions (such as second-order logic, situation calculus and first-order probabilistic logic) and variants (such as Datalog and first-order answer set programming) have been extensively studied in the literature [2,8].

Nevertheless, AI researchers have pointed out several issues regarding using FOL for the purpose of knowledge representation and reasoning, mostly from the reasoning point of view. For instance, FOL is computationally very difficult. Reasoning about FOL is a well-known undecidable problem. Also, FOL is monotonic in the sense that adding new knowledge into a first-order knowledge base will always result in more consequences. However, human reasoning is sometimes nonmonotonic.

In this paper, we argue that FOL also has some critical disadvantages merely from the knowledge representation point of view. First of all, although FOL is considered natural for well-trained logicians, it is not simple and flexible enough for knowledge engineers with less training. One possible reason is the distinction and hierarchy between term level (including constants, variables and terms), predicate level (including predicates and functions) and formula level (including

© Springer International Publishing AG 2017
T. Everitt et al. (Eds.): AGI 2017, LNAI 10414, pp. 87–97, 2017.
DOI: 10.1007/978-3-319-63703-7_9

atoms and compound formulas/sentences). From my own experience as a teacher in this subject, although strongly emphasized in the classes, many students failed to understand why a predicate or an atom cannot be in the scope of a function. Another reason is the notion of free occurrences of variables. For instance, it is not easily understandable for many students why the GEN inference rule has to enforce the variable occurrence restrictions. Last but not least, arbitrary nesting is another issue. Again, although natural from a mathematical point of view, a nested formula, e.g., $(x \vee \neg(y \wedge z)) \wedge (\neg y \vee \neg x)$ is hard to be understood and used.

Secondly, FOL has limitations in terms of expressive power. Because of the hierarchy from the term level to the formula level, FOL cannot quantify over predicates/functions. This can be addressed by extending FOL into high-order logic. Nevertheless, high-order logic still cannot quantify over formulas. As a consequence, FOL and high-order logic are not able to represent an axiom or an inference rule in logic, such as *Modus Ponens*. As an example, in automated solving mathematical problems, we often use proof by induction. To represent this, we need to state that for some statement P with a number parameter, if that P holds for all numbers less than k implies that P holds for the number k as well, then P holds for all natural numbers. Here, P is a statement at a formula level, possibly with complex sub-statements within itself. Hence, in order to represent proof by induction, we need to quantify over formulas.

Thirdly, FOL itself can hardly formalize some important notions including probability, actions, time etc., which are needed in a wide range of AI applications. For this purpose, AI researchers have made significant progresses on extending FOL with these notions separately, such as first-order probabilistic logic [1,7], situation calculus [9,10], CTL [3] etc. Each is a challenging task in the sense that it has to completely re-define the syntax as well as the semantics. However, combing these notions together, even several of them, seems an extremely difficult task. Moreover, there are many more building blocks to be incorporated. For instance, consider task planning for home service robots. It is necessary to represent and reason about actions, probability, time and more building blocks such as preferences altogether at the same time.

To address these issues, we propose assertional logic, in which all syntactic objects are categorized as set theoretic constructs including individuals, concepts and operators, and all kinds of knowledge are uniformly formalized by equality assertions of the form $a = b$, where a and b are either atomic individuals or compound individuals. Semantically, individuals, concepts and operators are interpreted as elements, sets and functions respectively in set theory and knowledge of the form $a = b$ means that the two individuals a and b are referring to the same element.

We first present the primitive form of assertional logic that uses minimal assumed knowledge and primitive constructs. Then, we show how to extend it with more building blocks by definitions, which are special kinds of knowledge, i.e., assertions used to define new individuals, concepts and operators. Once these new syntactic objects are defined, they can be used as a basis to define more. We show that assertional logic, although simpler, is more expressive and extensible

than FOL. As a case study, we show how to extend assertional logic for unifying logic and probability. Note that our intention is not to reinvent the wheel of these building blocks but to borrow existing excellent work on formalizing these building blocks separately and to assemble them within one framework (i.e., assertional logic) so that they can live happily ever after.

2 Assertional Logic: The Primitive Form

One cannot build something from nothing. Hence, in order to establish assertional logic, we need some basic knowledge. Of course, for the purpose of explanation, we need an informal meta language whose syntax and semantics are pre-assumed. As usual, we use a natural language such as English. Nevertheless, this meta language is used merely for explanation and it should not affect the syntax as well as the semantics of anything defined formally.

Only a meta level explanation language is not enough. Other than this, we also need some core objects and knowledge, whose syntax and semantics are pre-assumed as well. These are called *prior objects* and *prior knowledge*. For instance, when defining real numbers, we need some prior knowledge about natural numbers; when defining probability, we need some prior knowledge about real numbers.

In assertional logic, we always treat the equality symbol "=" as a prior object. There are some prior knowledge associated with the equality symbol. For instance, "=" is an equivalence relation satisfying reflexivity, symmetricity, and transitivity. Also, "=" satisfies the general substitution property, that is, if $a = b$, then a can be used to replace b anywhere. Other than the equality symbol, we also assume some prior objects and their associated prior knowledge in set theory [6], including set operators such as set union and Cartesian product, Boolean values, set builder notations and natural numbers.

Given an application domain, a *syntactic structure* (*structure* for short if clear from the context) of the domain is a triple $\langle \mathcal{I}, \mathcal{C}, \mathcal{O} \rangle$, where $\mathcal{I}$ is a collection of *individuals*, representing objects in the domain, $\mathcal{C}$ a collection of *concepts*, representing groups of objects sharing something in common and $\mathcal{O}$ a collection of *operators*, representing relationships and connections among individuals and concepts. Concepts and operators can be nested and considered as individuals as well. If needed, we can have concepts of concepts, concepts of operators, concepts of concepts of operators and so on.

An operator could be multi-ary, that is, it maps a tuple of individuals into a single individual. Each multi-ary operator O is associated with a *domain* of the form $(C_1, \ldots, C_n)$, representing all possible values that the operator O can operate on, where $C_i, 1 \leq i \leq n$, is a concept. We call n the *arity* of O. For a tuple $(a_1, \ldots, a_n)$ matching the domain of an operator O, i.e., $a_i \in C_i, 1 \leq i \leq n$, O maps $(a_1, \ldots, a_n)$ into an individual, denoted by $O(a_1, \ldots, a_n)$. We also use $O(C_1, \ldots, C_n)$ to denote the set $\{O(a_1, \ldots, a_n) \mid a_i \in C_i\}$, called the *range* of the operator O.

Operators are similar to functions in first-order logic but differs in two essential ways. First, operators are many-sorted as $C_1, \ldots, C_n$ could be different concepts. More importantly, $C_1, \ldots, C_n$ could be high-order constructs, e.g., concepts of concepts, concepts of operators.

For instance, consider a family relationship domain, in which *Alice* and *Bob* are individuals, $Human$, $Woman$ and $Female$ are concepts and $Father$, $Mother$ and *Aunt* are operators etc.

Let $\langle \mathcal{I}, \mathcal{C}, \mathcal{O} \rangle$ be a syntactic structure. A *term* is an individual, either an atomic individual $a \in \mathcal{I}$ or the result $O(a_1, \ldots, a_n)$ of an operator O operating on some individuals $a_1, \ldots, a_n$. We also call the latter *compound individuals*.

An *assertion* is of the form

$$a = b, \tag{1}$$

where a and b are two terms. Intuitively, an assertion of the form (1) is a piece of knowledge in the application domain, claiming that the left and right side refer to the same object.

A *knowledge base* is a set of assertions. Terms and assertions can be considered as individuals as well. For instance, in the family relationship domain, $Father(Alice) = Bob$, $Father(Alice) = Uncle(Bob)$ are assertions.

Similar to concepts that group individuals, we use schemas to group terms and assertions. A *schema term* is either an atomic concept $C \in \mathcal{C}$ or of the form $O(C_1, \ldots, C_n)$, where $C_i, 1 \leq i \leq n$ are concepts. Essentially, a schema term represents a set of terms, in which every concept is grounded by a corresponding individual. For instance, $O(C_1, \ldots, C_n)$ is the collection $\{O(a_1, \ldots, a_n)\}$, where $a_i \in C_i, 1 \leq i \leq n$ are individuals. Then, a *schema assertion* is of the same form as form (1) except that terms can be replaced by schema terms. Similarly, a schema assertion represents a set of assertions.

We say that a schema term/assertion *mentions* a set $\{C_1, \ldots, C_n\}$ of concepts if $C_1, \ldots, C_n$ occur in it, and *only mentions* if $\{C_1, \ldots, C_n\}$ contains all concepts mentioned in it. Note that it could be the case that two or more different individuals are referring to the same concept C in schema terms and assertions. In this case, we need to use different *copies* of C, denoted by $C^1, C^2, \ldots$, to distinguish them. For instance, all assertions $x = y$, where x and y are human, are captured by the schema assertion $Human^1 = Human^2$. On the other side, in a schema, the same copy of a concept C can only refer to the same individual. For instance, $Human = Human$ is the set of all assertions of the form $x = x$, where $x \in Human$.

We introduce a set theoretic semantics for assertional logic. Since we assume set theory as the prior knowledge, in the semantics, we freely use those individuals (e.g., the empty set), concepts (e.g., the set of all natural numbers) and operators (e.g., the set union operator) without explanation.

An *interpretation* (also called a *possible world*) is a pair $\langle \Delta, .^I \rangle$, where Δ is a domain of elements, and $.^I$ is a mapping function that admits all prior knowledge, and maps each individual into a domain element in Δ, each concept into a set in Δ and each n-ary operator into an n-ary function in Δ. The mapping function $.^I$ is generalized for terms as well by mapping $O(a_1, \ldots, a_n)$

to $O^I(a_1^I, \ldots, a_n^I)$. Similar to terms and assertions, interpretations can also be considered as individuals to be studied.

It is important to emphasize that an interpretation has to admit all the prior knowledge. For instance, since we assume set theory, suppose that an interpretation maps two individuals x and y as the same element a in the domain, then the concepts $\{x\}$ and $\{y\}$ must be interpreted as $\{a\}$, and $x = y$ must be interpreted as $a = a$.

Let I be an interpretation and $a = b$ an assertion. We say that I is a *model* of $a = b$, denoted by $I \models a = b$ iff $.^I(a) = .^I(b)$, also written $a^I = b^I$. Let KB be a knowledge base. We say that I is a model of KB, denoted by $I \models KB$, iff I is a model of all assertions in KB. We say that an assertion A is a *property* of KB, denoted by $KB \models A$, iff all models of KB are also models of A. In particular, we say that an assertion A is a *tautology* iff it is modeled by all interpretations.

Since we assume set theory as our prior knowledge, we directly borrow some set theoretic constructs. For instance, we can use $\cup(C_1, C_2)$ (also written as $C_1 \cup C_2$) to denote a new concept that unions two concepts C_1 and C_2. Applying this to assertions, we can see that assertions of the primitive form (1) can indeed represent many important features in knowledge representation. For instance, the *membership assertion*, stating that an individual a is an instance of a concept C is the following assertion $\in (a, C) = \top$ (also written as $a \in C$). The *containment assertion*, stating that a concept C_1 is contained by another concept C_2, is the following assertion $\subseteq (C_1, C_2) = \top$ (also written as $C_1 \subseteq C_2$). The *range declaration*, stating that the range of an operator O operating on some concept C_1 equals to another concept C_2 is the following assertion $O(C_1) = C_2$.

3 Extending New Syntactic Objects by Definitions

As argued in the introduction section, extensibility is a critical issue for knowledge representation and modeling. In assertional logic, we use *definitions* for this purpose. Definitions are (schema) assertions used to define new syntactic objects (including individuals, concepts and operators) based on existing ones. Once these new syntactic objects are defined, they can be used to define more. Note that definitions are nothing extra but special kinds of knowledge (i.e. assertions).

We start with defining new individuals. An individual definition is an assertion of the form

$$a = t, \tag{2}$$

where a is an atomic individual and t is a term. Here, a is the individual to be defined. This assertion claims that the left side a is defined as the right side t. For instance, $0 = \emptyset$ means that the individual 0 is defined as the empty set.

Defining new operators is similar to defining new individuals except that we use schema assertions instead. Let O be an operator to be defined and $(C_1, \ldots, C_n)$ its domain. An operator definition is a schema assertion of the form

$$O(C_1, \ldots, C_n) = T, \tag{3}$$

where T is a schema term that mentions concepts only from $C_1, \ldots, C_n$. It could be the case that T only mentions some of $C_1, \ldots, C_n$. Note that if $C_1, \ldots, C_n$ refer to the same concept, we need to use different copies.

Since a schema assertion represents a set of assertions, essentially, an operator definition of the form (3) defines the operator O by defining the value of $O(a_1, \ldots, a_n)$ one-by-one, where $a_i \in C_i, 1 \leq i \leq n$. For instance, for defining the successor operator $Succ$, we can use the schema assertion $Succ(\mathbb{N}) = \{\mathbb{N}, \{\mathbb{N}\}\}$, meaning that, for every natural number n, the successor of n, is defined as $\{n, \{n\}\}$, i.e., $Succ(n) = \{n, \{n\}\}$.

Defining new concepts is somewhat different. As concepts are essentially sets, we directly borrow set theory notations to define concepts. There are four ways to define a new concept.

Enumeration. Let $a_1, \ldots, a_n$ be n individuals. Then, the collection $\{a_1, \ldots, a_n\}$ is a concept, written as

$$C = \{a_1, \ldots, a_n\}. \tag{4}$$

For instance, we can define the concept $Digits$ by $Digits = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Operation. Let C_1 and C_2 be two concepts. Then, $C_1 \cup C_2$ (the union of C_1 and C_2), $C_1 \cap C_2$ (the intersection of C_1 and C_2), $C_1 \backslash C_2$ (the difference of C_1 and C_2), $C_1 \times C_2$ (the Cartesian product of C_1 and C_2), 2^{C_1} (the power set of C_1) are concepts. Operation can be written by assertions as well. For instance, the following assertion

$$C = C_1 \cup C_2 \tag{5}$$

states that the concept C is defined as the union of C_1 and C_2. As an example, one can define the concept Man by $Man = Human \cap Male$.

Comprehension. Let C be a concept and $A(C)$ a schema assertion that only mentions concept C. Then, individuals in C satisfying A, denoted by $\{x \in C | A(x)\}$ (or simply $C|A(C)$), form a concept, written as

$$C' = C|A(C). \tag{6}$$

For instance, we can define the concept $Male$ by $Male = \{Animal \mid Sex(Animal) = male\}$, meaning that $Male$ consists of all animals whose sexes are male.

Replacement. Let O be an operator and C a concept on which O is well defined. Then, the individuals mapped from C by O, denoted by $\{O(x) \mid x \in C\}$ (or simply $O(C)$), form a concept, written as

$$C' = O(C). \tag{7}$$

For instance, we can define the concept $Parents$ by $Parents = ParentOf(Human)$, meaning that it consists of all individuals who is a $ParentOf$ some human.

Definitions can be incremental. We may define some syntactic objects first. Once defined, they can be used to define more. One can always continue with this incremental process. For instance, in arithmetic, we define the successor operator first. Once defined, it can be used to define the add operator, which is further served as a basis to define more useful syntactic objects.

For clarity, we use the symbol "::=" to replace "=" for definitions.

4 Embedding Classical Logic into Assertional Logic

In the previous section, we show how to extend assertions of the primitive form (1) into multi-assertions and nested assertions. In this section, we continue with this task to show how to define more complex forms of assertions with logic connectives, including not only propositional connectives but also quantifiers.

We start with the propositional case. Let $\mathcal{A}$ be the concept of nested assertions. We introduce a number of operators over $\mathcal{A}$ in assertional logic, including $\neg(\mathcal{A})$ (for *negation*), $\wedge(\mathcal{A}^1, \mathcal{A}^2)$ (for *conjunction*), $\vee(\mathcal{A}^1, \mathcal{A}^2)$ (for *disjunction*) and $\rightarrow(\mathcal{A}^1, \mathcal{A}^2)$ (for *implication*).

There could be different ways to define these operators in assertional logic. Let $a = a'$ and $b = b'$ be two (nested) assertions. The propositional connectives are defined as follows:

$$\begin{aligned}
\neg(a = a') &::= \{a\} \cap \{a'\} = \emptyset \\
\wedge(a = a', b = b') &::= (\{a\} \cap \{a'\}) \cup (\{b\} \cap \{b'\}) = \{a, a', b, b'\} \\
\vee(a = a', b = b') &::= (\{a\} \cap \{a'\}) \cup (\{b\} \cap \{b'\}) \neq \emptyset \\
\rightarrow(a = a', b = b') &::= (\{a, a'\} \setminus \{a\} \cap \{a'\}) \cup (\{b\} \cap \{b'\}) \neq \emptyset.
\end{aligned}$$

We also use $a \neq a'$ to denote $\neg(a = a')$. One can observe that the ranges of all logic operators are nested assertions. Hence, similar to multi-assertion and nested assertion, propositional logic operators are syntactic sugar as well in assertional logic.

It can be observed that all tautologies in propositional logic (e.g., De-Morgan's laws) are also a tautology in assertional logic in the sense that each proposition is replaced by an assertion and each propositional connective is replaced by corresponding logic operators in assertional logic.

Now we consider to define operators for quantifiers, including $\forall$ (for the *universal* quantifier) and $\exists$ (for the *existential* quantifier). The domain of quantifiers is a pair $(C, A(C))$, where C is a concept and $A(C)$ is a schema assertion that only mentions C.

The quantifiers are defines as follows:

$$\forall(C, A(C)) ::= C | A(C) = C \tag{8}$$

$$\exists(C, A(C)) ::= C | A(C) \neq \emptyset \tag{9}$$

Intuitively, $\forall(C, A(C))$ is true iff those individuals x in C such that $A(x)$ holds equals to the concept C itself, that is, for all individuals x in C, $A(x)$ holds; $\exists(C, A(C))$ is true iff those individuals x in C such that $A(x)$ holds does not equal to the empty set, that is, there exists at least one individual x in C such

that $A(x)$ holds. We can see that the ranges of quantifiers are nested assertions as well. Thus, quantifiers are also syntactic sugar of the primitive form.

Note that quantifiers defined here are ranging from an arbitrary concept C. If C is a concept of all atomic individuals and all quantifiers range from the same concept C, then these quantifiers are first-order. Nevertheless, the concepts could be different. In this case, we have many-sorted first-order logic. Moreover, C could be complex concepts, e.g., a concept of all possible concepts. In this case, we have monadic second-order logic. Yet C could be many more, e.g., a concept of assertions, a concept of concepts of terms etc. In this sense, the quantifiers become high-order. Finally, the biggest difference is that C can even be a concept of assertions so that quantifiers in assertional logic can quantify over assertions (corresponding to formulas in classical logics), while this cannot be done in classical logics.

A problem arises whether there is cyclic definition as we assume first-order logic as our prior knowledge. Nevertheless, although playing similar roles, operators (over assertions) defined in assertional logic are considered to be different from logic connectives (over propositions/formulas) since they are on a different layer of definition. The main motivation is for the purpose of extensibility, i.e., by embedding classical logic connectives into operators in assertional logic, we can easily extend it with more components and building blocks including probability.

5 Incorporating Probability

Probability is another important building block for knowledge representation and modeling. In the last several decades, with the development of uncertainty in artificial intelligence, a number of influential approaches [1,4,5,11–13] have been developed, and important applications have been found in machine learning, natural language processing etc.

In this section, we show how logic and probability can be unified through assertions in assertional logic. The basic idea is that, although the interactions between logic and probability are complicated, their interactions with assertions of the form (1) could be relatively easy. As shown in the previous section, the interactions between logic and assertions can be defined by a few lines. In this section, following Gaifman's idea [4], we show that this is indeed the case for integrating assertions with probability as well. As a result, the interactions between logic and probability will be automatically established via assertions.

Since operations over real numbers are involved in defining probability, we need to assume a theory of real number as our prior knowledge.

Gaifman [4] proposed to define the probability of a logic sentence by the sum of the probabilities of the possible worlds satisfying it. Following this idea, in assertional logic, we introduce an operator Pr (for probability) over the concept $\mathcal{A}$ of assertions. The range of Pr is the set of real numbers. For each possible world w, we assign an associated weight W_w, which is a positive real number. Then, for an assertion A, the probability of A, denoted by $Pr(A)$, is define by the following schema assertion:

$$Pr(A) = \frac{\Sigma_{w,w\models A}\ W_w}{\Sigma_w\ W_w}. \tag{10}$$

This definition defines the interactions between probability and assertions. In case that there are a number of infinite worlds, we need to use measure theory. Nevertheless, this is beyond the scope of our paper, which focuses on how to use assertional logic for extensible knowledge modeling.

Once we have defined the probability $Pr(A)$ of an assertion A as a real number, we can directly use it inside other assertions. In this sense, $Pr(A) = 0.5$, $Pr(A) \geq 0.3$, $Pr(A) \geq Pr(\forall(C, B(C)))-0.3$, $Pr(A)\times 0.6 \geq 0.4$ and $Pr(Pr(A) \geq 0.3) \geq 0.3$ are all valid assertions. We are able to investigate some properties about probability, for instance, Kolmogorov's first and second probability axioms, that is, (1) for all assertions, $Pr(A) \geq 0$, and (2) if A is a tautology, then $Pr(A) = 1$.

We also extend this definition for conditional probability. We again introduce a new operator Pr over pairs of two assertions. Following a similar idea, the conditional probability $Pr(A_1, A_2)$ of an assertion A_1 providing another assertion A_2, also denoted by $Pr(A_1|A_2)$, is defined by the following schema assertion:

$$Pr(A_1|A_2) = \frac{\Sigma_{w,w\models A_1, w\models A_2}\ W_w}{\Sigma_{w,w\models A_2}\ W_w}. \tag{11}$$

Again, once conditional probability is defined as a real number, we can use it arbitrarily inside other assertions. Similarly, we can derive some properties about conditional probabilities, including the famous Bayes' theorem, i.e.,

$$Pr(A_1) \times Pr(A_2|A_1) = Pr(A_2) \times Pr(A_2)Pr(A_1|A_2).$$

for all assertions A_1 and A_2.

Although we only define probabilities for assertions of the basic form, the interactions between probability and other building blocks, e.g., logic, are automatically established since assertions connected by logic operators can be reduced into the primitive form. In this sense, we can investigate some properties about the interactions between logic and probability. For instance, it can be observed that Kolmogorov's third probability axiom is a tautology in assertion logic. That is, let $A_1, \ldots, A_n$ be n assertions that are pairwise disjoint. Then, $Pr(A_1 \vee \cdots \vee A_n) = Pr(A_1) + \cdots + Pr(A_n)$.

It can be verified that many axioms and properties regarding the interactions between logic and probability are tautologies in assertional logic as well, for instance, the additivity axiom: $Pr(\phi) = Pr(\phi \wedge \psi) + Pr(\phi \wedge \neg\psi)$ and the distributivity axiom: $\phi \equiv \psi$ implies that $Pr(\phi) = Pr(\psi)$, for any two assertions ϕ and ψ. In this sense, assertional logic can also be used to validate existing properties about the interactions of logic and probability. In addition, it may foster new discoveries, e.g., the interactions between higher-order logic and probability and some properties about nested probabilities.

6 Discussion and Conclusion

In this paper, we argue that, for the purpose of knowledge representation, classical first-order logic has some critical issues, including simplicity, flexibility, expressivity and extensibility. To address these issues, we propose assertional logic instead, in which the syntax of an application domain is captured by individuals (i.e., objects in the domain), concepts (i.e., groups of objects sharing something in common) and operators (i.e., connections and relationships among objects), and knowledge in the domain is simply captured by equality assertions of the form $a = b$, where a and b are terms.

In assertional logic, without redefining the semantics, one can extend a current system with new syntactic objects by definitions, which are special kinds of knowledge (i.e., assertions). Once defined, these syntactic objects can be used to define more. This can be done for assertional logic itself. We extend the primitive form of assertional logic with logic connectives and quantifiers. The key point is that, when one wants to integrate a new building block in assertional logic, she only needs to formalize it as syntactic objects (including individuals, concepts and operators) and defines its interactions with the basic form of assertions (i.e., $a = b$). The interactions between this building block and others will be automatically established since all complicated assertions can essentially be reduced to the basic form. As a case study, we briefly discuss how to incorporate probability in this paper.

Of course, assertional logic is deeply originated from first-order logic. Individuals, concepts and operators are analogous to constants, unary predicates and functions respectively, and assertion is inspired by equality atom. Nevertheless, they differ from many essential ways. Firstly, individuals can be high-order objects, e.g., concepts and assertions, so are concepts and operators. Secondly, assertional logic is naturally many-sorted, that is, the domain of an operator can be a tuple of many different concepts including high-order ones. Thirdly, concepts play a central role in assertional logic, which is natural for human knowledge representation. While concepts can be formalized as unary predicates in FOL, they are not specifically emphasized. Fourthly, in assertional logic, all kinds of knowledge are uniformly formalized in the same form of equality assertions. As shown in Sect. 5, complicated logic sentences are defined as equality assertions as well by embedding connectives and quantifiers as operators over assertions. Fifthly, following the above, although connectives, quantifiers and nesting can be represented in assertional logic, they are not considered as primitive constructs. In this sense, they will only be used on demand when necessarily needed. For instance, each uses of nesting essentially introduces a new syntactic object. We argue that this is an important reason that makes assertional logic simpler than FOL. Sixthly, in assertional logic, the simple form of $a = b$ is expressive as a and b can be high-order constructs and can be inherently related within a rich syntactic structure. While in FOL, an equality atom does not have this power. Last but not least, assertional logic directly embraces extensibility within its own framework by definitions. For instance, to define quantifiers, assertional

logic only needs two lines (see Eqs. 8 and 9) without redefining a whole new syntax and semantics, which is much simpler than FOLs.

This paper is only concerned with the representation task and the definition task, and we leave the reasoning task to our future work. Nevertheless, we argue that representation and definition are worth study on their own merits. Such successful stories include entity-relationship diagram, semantic network and many more. Besides, extending assertional logic with some important AI building blocks, e.g., actions and their effects, is challenging and worth pursuing.

Acknowledgement. The author would like to thank Fangzhen Lin for his valuable comments on this paper.

References

1. Bacchus, F.: Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities. MIT Press, Cambridge (1990)
2. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier, Amsterdam (2004)
3. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982). doi:10.1007/BFb0025774
4. Gaifman, H.: Concerning measures in first order calculi. Israel J. Math. **2**, 1–18 (1964)
5. Hailperin, T.: Probability logic. Notre Dame J. Formal Logic **25**, 198–212 (1984)
6. Halmos, P.: Naive Set Theory. Undergraduate Texts in Mathematics. Springer, New York (1974)
7. Halpern, J.Y.: An analysis of first-order logics of probability. Artif. Intell. **46**(3), 311–350 (1990)
8. van Harmelen, F., Lifschitz, V., Porter, B.W. (eds.): Handbook of Knowledge Representation. Foundations of Artificial Intelligence, vol. 3. Elsevier, Amsterdam (2008)
9. Levesque, H., Pirri, F., Reiter, R.: Foundations for the situation calculus. Electron. Trans. Artif. Intell. **2**(3–4), 159–178 (1998)
10. Lin, F.: Situation calculus. In: Handbook of Knowledge Representation, pp. 649–669 (2008)
11. Milch, B.C.: Probabilistic models with unknown objects. Ph.D. thesis, Berkeley, CA, USA, aAI3253991 (2006)
12. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
13. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)