International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:2, No:1, 2008

Reasoning with Dynamic Domains and Computer Security

Yun Bai

***Abstract*—** Representing objects in a dynamic domain is essential in commonsense reasoning under some circumstances. Classical logics and their nonmonotonic consequences, however, are usually not able to deal with reasoning with dynamic domains due to the fact that every constant in the logical language denotes some existing object in the static domain. In this paper, we explore a logical formalization which allows us to represent nonexisting objects in commonsense reasoning. A formal system named N-theory is proposed for this purpose and its possible application in computer security is briefly discussed.

***Keywords*—** knowledge representation and reasoning, commonsense reasoning, computer security

I. Introduction

Classical logics and their nonmonotonic consequences have been used as major formal methods in commonsense reasoning, knowledge representation, robotic dynamics modeling and computer security. From a semantical point of view, a common feature of these logics is that every syntactic term in the language represents some real object in the problem domain. For instance, an atom $Bird(Tweety)$ represents a fact that there is an object in our domain named *Tweety* which is a bird. However, in some circumstances, it may be needed to represent some knowledge that involves nonexisting objects in our domain. For instance, a computer security domain to deal with a not-yet joined user or application. As it will be shown next, current logics have difficulties in dealing with this kind of knowledge in reasoning.

Example 1: **The flying horse paradox**. We consider a statement like

$$\textit{all existing objects are not flying horses.} \tag{1}$$

If we formalize this statement in first order logic, we may probably write a sentence as follows:

$$\forall x \exists y (y = x \supset \neg Flying\text{-}horse(x)). \tag{2}$$

Now we consider another statement that

$$\textit{Pegasus is a flying horse}. \tag{3}$$

Obviously we may write a sentence

$$Flying\text{-}horse(Pegasus)$$

to represent statement (3) where *Pegasus* is supposed to be a constant in our language.

However, sentence $Flying\text{-}horse(Pegasus)$ is actually *inconsistent* with (2). We note that formula $\forall x \exists y (y = x)$ is a theorem in any first order theory [6]. So (2) can be reduced to

$$\forall x \neg Flying\text{-}horse(x). \tag{4}$$

Then from *Substitution Axiom*

$$\forall x \varphi \supset \varphi_x[\mathbf{a}] \tag{5}$$

and (4), we can derive

$$\neg Flying\text{-}horse(Pegasus). \tag{6}$$

But semantically, statement (3) does not really conflict with statement (1) because we do not declare that $Pegasus$ is an existing object. Furthermore, if we assume that $Pegasus$ be a constant denoting some nonexisting object, epsitemically, we should be able to state that *Pegasus is a flying horse* although we have already explicitly stated that *all existing objects are not flying horses*. Clearly, classical first order logic can not represent nonexisting objects since every constant/term in the language must be mapped to some real object in the domain. ■

Representing nonexisting objects is important in computer security. For instance, a user may be required to be deleted or added to the system. These kinds of actions are generally difficult to represent within current formal theories of dynamic modeling.

In this paper, we explore a logical formalization which allows us to represent nonexisting objects in an agent's knowledge base. The paper is organized as follows. Section 2 proposes a formal system named N-theory where sentences involving constants that denote nonexisting objects are allowed to be presented. Section 3 discusses its semantics and property. Section 4 investigates its possible application in computer security. Finally, section 5 concludes the paper with some remarks.

II. N-Theory

The formal system we consider in this paper consists of the following components:

- The *language* is an arbitrary first order language $\mathcal{L}$ without function symbols. We also use **a, b, c** $\cdots$ to denote metavariables for constants[1].
- The *logical axioms* are:
  - (A1) φ, where φ is a tautology (Propositional Axiom),
  - (A2) $(\forall x \varphi \wedge \exists x(x = \mathbf{a})) \supset \varphi_x[\mathbf{a}]$ (Restricted Substituition Axiom),
  - (A3) $\mathbf{a} = \mathbf{a}$ (Identity Axiom 1),
  - (A4) $\mathbf{a} = \mathbf{b} \supset (\varphi(\mathbf{a}) \supset \varphi(\mathbf{b}))$ (Identity Axiom 2).

Yun Bai is with School of Computing and Mathematics, University of Western Sydney, NSW 1797, Australia, E-mail: ybai@scm.uws.edu.au

[1] $\varphi(\mathbf{a})$ is called a *sentence scheme* which is viewed as a set of sentences $\{\varphi(a) \mid \ a$ is any constant of $\mathcal{L}\}$.

- The *inference rules* are:
  (I1) Infer ψ from φ and $\varphi \supset \psi$ (Modus Ponens),
  (I2) Infer $\forall x\varphi$ from $\exists x(x = \mathbf{a}) \supset \varphi_x[\mathbf{a}]$ (Restricted Universal Generalization).

Note that (A2) and (I2) represent the major difference between our system and standard first order logic. Condition $\exists x(x = \mathbf{a})$ indicates that constant $\mathbf{a}$ denotes some existing object, which is needed in (A2) and (I2) to guarantee that a universally quantified sentence can only be substituted by or generalized from its *existing* instances. A N-*theory* is a pair $<T, C>$, where T is a set of sentences of $\mathcal{L}$ and C is a nonempty set of constants of $\mathcal{L}$. The logical axioms of $<T, C>$ are (A1)-(A4) and inference rules of $<T, C>$ are (I1) and (I2). Furthermore, $<T, C>$ also has following two *domain-independent nonlogical axioms*:

(A5) $\exists x(x = \mathbf{a})$, where $\mathbf{a} \in C$ (Denotation Axiom),
(A6) for any $\mathbf{a} \in C$ and $\mathbf{b} \notin C$, $\mathbf{a} \neq \mathbf{b}$ (Disjoint Name Axiom).

Definition 1: A sentence φ is N-*free* (i.e. nonexisting object free) with respect to a N-theory $<T, C>$ if φ is not a logical or domain-independent nonlogical axiom, and

(i) for any non-equality predicate symbol P occurring in φ, each constant c occurring in $P(\cdots)$ is in C;
(ii) for any equality $=$ occurring in φ, there is at most one constant not belonging to C taken by $=$[2].

If we view a N-theory $<T, C>$ as an agent's knowledge base, then each constant specified in C denotes some existing object in the agent's domain while all other constants not in C denote nonexisting objects with respect to this domain. On the other hand, a N-free sentence $<T, C>$ represents some properties on existing objects in the agent's domain. Each N-free sentence has a clear semantics under the interpretation of $<T, C>$. Note that a N-theory may also include sentences that are not N-free with respect to this theory. If a N-free sentence φ with respect to $<T, C>$ is also included in T, we say that φ is a N-free sentence of $<T, C>$.

Besides logical axioms (A1)-(A4) and domain-independent nonlogical axioms (A5) and (A6), all other sentences in $<T, C>$ are called *domain-dependent* nonlogical axioms. In the rest of the paper, whenever there is no confusion in the context, we will only present domain-dependent nonlogical axioms in a N-theory. It is clear that if C is the set of all constants of $\mathcal{L}$, then N-theory $<T, C>$ is reduced to a classical first order theory.

Definition 2: Given a N-theory $<T, C>$ and a sentence φ. φ is a N-*theorem* of $<T, C>$, denoted as $<T, C> \vdash_N \varphi$, if

(i) φ is a logical axiom or a domain-independent nonlogical axiom;
(ii) φ is a N-free sentence of $<T, C>$;
(iii) φ is the conclusion of an inference rule (I1) or (I2) where all of the hypotheses of the rule are also N-theorems of $<T, C>$.

$<T, C>$ is N-*inconsistent* if there exists some sentence φ such that $<T, C> \vdash_N \varphi$ and $<T, C> \vdash_N \neg\varphi$; otherwise $<T, C>$ is N-*consistent*.

[2]For instance, if $a = b$ occurs in φ, then at most one of a and b does not belong to C.

Example 2: Consider a N-theory $<T, \{Prancer\}>$ where $T = \{Horse(Pegasus), Horse(Prancer), \forall x Horse(x) \supset Runfast(x)\}$.
Since $Horse(Pegasus)$ is not a N-free sentence of this N-theory, we have $<T, \{Prancer\}> \nvdash_N Runfast(Pegasus)$ and
$<T, \{Prancer\}> \vdash_N Runfast(Prancer)$. ■

III. Semantics and Properties of N-Theory

In this section, we briefly investigate the semantics of N-theory and its properties. We propose non-classical structures named N-structures for the language of N-theory where some constants are allowed to denote nonexisting objects with respect to the domain of quantification of the structure.

Definition 3: A N-*structure* M of $\mathcal{L}$ is any ordered pair $(D, \mathcal{F})$, where D is a set (to be called the *domain*) and $\mathcal{F}$ is a unary function such that:

(i) $\mathcal{F}$ is total to assign every n-placed predicate symbol P a set of ordered n-placed tuples of elements of D;
(ii) $\mathcal{F}$ is partial to assign every *defined* constant an element of D.

Note that a N-structure M is usually associated with a set of constants of $\mathcal{L}$ where each constant in the set is defined in M (i.e. mapped to some element in the domain of M) and all other constants of $\mathcal{L}$ are then undefined in M. Clearly, if the set of defined constants is identical to the set of all constants of $\mathcal{L}$, a N-structure is reduced to a classical first order structure of $\mathcal{L}$.

We define a model for a N-theory.

Definition 4: Given a N-theory $<T, C>$. M is a N-*model* of $<T, C>$ if

(i) every constant in C is defined in M and all other constants of $\mathcal{L}$ are not defined in M;
(ii) every N-free sentence of $<T, C>$ is N-satisfied in M.

A sentence φ is N-*entailed* by $<T, C>$, denoted as $<T, C> \models_N \varphi$, if φ is N-satisfied in every N-model of $<T, C>$.

Now we discuss the basic properties of N-theory. Given a N-theory $<T, C>$ and a N-free sentence φ relating to $<T, C>$. From Definition 1, it is observed that the only possibility that some constant $b \notin C$ occurs in φ is in the form of $b = b$, $a = b$ or $x = b$ where $a \in C$. Whenever φ is a N-free relating to a N-theory $<T, C>$, we can equivalently view φ to its corresponding φ' in which every constant belongs to C. For example, if constant $b \notin C$, the N-free sentence $\forall x(P(x) \vee x = b)$ is equivalent to $\forall x P(x)$. Therefore, we assume that for any N-free sentence φ wrt $<T, C>$, every constant occurring in φ belongs to C.

IV. Application in Computer Security

Theoretically, any technique used to specify system security should provide a language with sufficient expressiveness to support the specification of complex concepts and procedures but it should also accompany this expressiveness with flexibility and mathematical precision and conciseness. A variety of logic security approaches [3], [5], [7] have been proposed for computer security since logic language has powerful expressiveness with precise syntax and semantics.

The security domain of a computer system contains security rules to protect the system from malicious attempts. A security domain for a real-time computer system is dynamic in the sense that the security rules need to be up to date, and be able to handle certain exceptions. For instance, when a new user or application or a new authorization right adds to the system, security rules regarding these new objects need to be flexible and powerful enough to process and reasoning under new circumstance.

In a security domain, we use $access(a)$ to denote that a is a constant object and has access right to the system. Currently if we have objects a, b, c in a category group and all the objects from this group have access right to the system, we should have $access(a)$, $access(b)$ and $access(c)$. If there's an exception that a cannot access the system, or the query asks the access right of a non existing object d, how can we handle these?

We propose to apply the N-theory just discussed to the security domain and to reason about it. We believe the N-theory can be properly justified to such a dynamic domain and to solve the issue of deleting and adding new object to the security domain.

V. Conclusion

In this paper, we presented the dynamic domain in logic reasoning, especially representation and reasoning with nonexisting objects. Although the issue of nonexisting objects has been studied by some philosophers, e.g. [2], with different motivation and background, however, their approaches seem hard to be used for our purpose in knowledge representation and reasoning. We also note that the null-value concept investigated in relational database theory [1] is related to represent nonexisting objects. But no proper logical formalization was proposed to deal with this problem in database area.

We plan to investigate the application of reasoning in dynamic domain in computer security. We will discuss how access rights are represented, and use the N-theory presented in this paper to handle reasoning in a dynamic security domain.

References

[1] P. Atzeni and V. de Antonellis, *Relational Database Theory*, The Benjamin/Cummings Publishing Company, Inc., 1993.
[2] E. Bencivenga, Free logic. *Handbook of Philosophical Logic*, Vol. III, pp373-426, 1986.
[3] E. Bertino, F. Buccafurri, E. Ferrari and P. Rullo, "A Logic-based Approach for Enforcing Access Control". *Computer Security*, vol.8, No.2-2, pp109–140, 2000.
[4] A. Herzig, J. Lang and P. Marquis, Action representation and partially observable planning using epistemic logic. *Proceedings of IJCAI03*, 1067–1072. 2003.
[5] N. Li, B. Grosof and J. Feigenbaum, "Delegation Logic: A Logic-based Approach to Distributed Authorization". *ACM Transactions on Information and System Security*, Vol.6, No.1, pp128–171, 2003.
[6] J. Shoenfield, *Mathematical Logic*. Addison-Wesley. 1967.
[7] L. Wang, D. Wijesekera and S. Jajodia, "A logic-based framework for attribute based access control," *Proceedings of the ACM Workshop on Formal Methods in Security Engineering*, pp45–55, 2004.