SDF: A Sign Description Framework for Cross-context Information Resource Representation and Interchange

Jingzhi Guo
Department of Computer and Information Science, Faculty of Science and Technology
University of Macau
Taipa, Macau
e-mail: jzguo@umac.mo

***Abstract*— While Internet is experiencing a drastic change, semantic representation study becomes more and more important in electronic business and enterprise systems. To semantically represent information resources, this paper proposes a novel Sign Description Framework, which is an abstract specification that represents any object in reality as signs. Based on a quaternary model of sign in semiotics, the proposed framework has several advantages such as non-ambiguity of resource representation in meaning, easy resource modeling, and easy cross-context heterogeneous resource mapping.**

Keywords-sign; sign description framework; SDF; cross-context; information resource representation; information exchange; quaternary model of sign

I. INTRODUCTION

Internet is experiencing a drastic change when it is more applied in electronic business (e-business). In this change, the intensive user participation asks for the study of e-business semantics in business communications on enterprise information systems [1], [7]. *Semantics* can be defined as the machine-computable, human-understandable, and program-reasonable meanings of concepts [5]. It is one of the most important tasks of Semantic Web [2] and Internet of Things [6]. It emphasizes on the meaning interchange between various information resources. One of its key research issues is the semantic representation of cross-context information resources to disambiguate the senses of human and computing devices for correct meaning interpretation during communication. Here, the cross-context information resource is a general term of everything heterogeneously formed and digitally represented. It can be defined as any digital object that refers to any object in reality of real or virtual, physical or spiritual, true or fictitious, and of abstract or concrete.

The research on the issue of semantic representation of cross-context information resources is important. First, it helps us understand the essence of everything relating to human interpretation of our world and its imaged digital world. Second, it aids us to represent everything as digital object. Third, it assists us to establish correct communication channels between context-different human groups and computing systems by using digital objects.

To conduct semantic representation research, this paper aims at developing a framework, called Sign Description Framework (SDF), in which any cross-context information resource is appropriately represented as digital object to contain interpretable meaning. In this paper, the SDF approach to semantic representation is concept-oriented, that is, it treats everything in the world as a meaningful concept referring to an object in reality. It treats a concept as a semiotic sign storable in computing systems. This sign reflects a digitally represented concept, referring to any object in reality, such as a term, a message, a picture, a video, a sound, an activity, a process, and a service. SDF defines a sign in a simple tree data model with abstract syntax and formal semantics according to a sign theory of semiotics [4], [8], [9]. It rigorously defines the notion of sign for well-founded deduction on SDF data.

The development of SDF is motivated by several aspects:

(1) *Computer-human-understandable information resource*. Any information resource shall be understandable by both computer and human, realizing the interpretation of information resources in a messaging cycle of human, computer, computer and human.
(2) *Consistent interpretation of information resource*. Any information resource shall have the same interpretation when it moves in any messaging cycle of human, computer, computer and human.
(3) *Automated information resource processing*. Any information resource shall be processed automatically across contexts.

The design of SDF is intended to meet the goals as follows: (1) using tree to construct simple data model; (2) representing objects in reality by signs; (3) having enumerable relations between signs; (4) using simple syntax for formal semantics; (5) enable reasoning between signs; and (6) allowing anyone to collaboratively create or modify signs.

In the rest of the paper, Section II introduces the sign theory. Section III proposes Sign Description Framework. In Section IV, various types of signs are defined. Section V compares with the existing resource description frameworks. Finally, Section VI concludes the paper with future work.

II. SIGN THEORY

Saussure describes a sign in a dyadic model of semiotics, that is, a sign is a representation of a signifier (i.e. structure of sign or form) and a signified (i.e. concept of sign or meaning) [9]. Applying this model in computing systems, a

structure or signifier is used as a form for computer to describe and refer to an object in reality (i.e. a real-world object), and a concept or signified is used to express meaning for human being to describe and refer to the same object in reality. By dyadic model, a concept is conveyed by a structure. For example, a unique identifier "12345" is a structure and a meaning "the ball of fire in the sky that the Earth goes round, and that gives us heat and light" is a concept, forming a sign (12345, "the ball of fire in the sky that the Earth goes round, and that gives us heat and light"). The word "sun" can again be a structure conveying a concept "the ball of fire in the sky that the Earth goes round, and that gives us heat and light". Thus, we have two signs (12345, "the ball of fire in the sky that the Earth goes round, and that gives us heat and light") and (sun, "the ball of fire in the sky that the Earth goes round, and that gives us heat and light"), which are semantically equivalent. The former structure "12345" is for computer use and the latter structure "sun" is for human use, which all refer to a same meaning.

Differently, in Peirce's triadic model of sign [8], object (in reality), sign and interpretant are interrelated to define a sign. An interpretant is a concept or the sense made of a sign by an interpreter, who interprets an object in reality into a sense of a sign and uses the sign to refer to the object in reality. Thus, a sign takes an interpreted meaning of an object in reality as a sign concept, which is further conveyed in a sign structure that refers to that object in reality. For example, the sun in the sky is an object in reality, a sign is (sun, "the ball of fire in the sky that the Earth goes round, and that gives us heat and light"), and an interpretant is the meaning "the ball of fire in the sky that the Earth goes round, and that gives us heat and light" interpreted by an interpreter. These three aspects constitutes the Peirce's triadic model of sign.

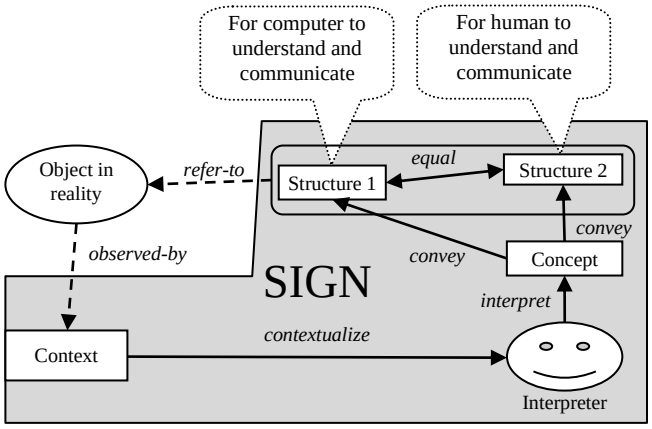

Figure 1. Sign Theory

In Guo's research on sign [4], Guo thought that there are many interpreters and each of them may be situated in different contexts (i.e., different semantic communities). Hence, a same object in reality may be interpreted in different ways. Thus, a concept or the sense made of a sign is a contextual interpretation of an object in reality in a particular context. Common concept of a sign is only available when all interpreters of a same object in reality share a common context and make a unanimous agreement on that concept. Inherited from Saussure's dyadic model and Peirce's triadic model, Guo added context element in representing any object in reality. This addition suggests a new sign model of (object in reality, (structure, concept), interpreter, context), which is a quaternary model of sign, where an interpreter in a context observes and interprets an object in reality into a concept conveyed in a structure to refer to the object in reality.

To enable both computer and human to interpret and understand a sign. This paper further divides structure into two types: computer-understandable structures (structure 1 shown in Figure 1) and human-understandable structures (structure 2 shown in Figure 1). This division allows us to map human understanding onto computer understanding and derives a semantic equivalent relationship between computer understandable structure and human understandable structure. The quaternary model of sign is the theoretical foundation of the SDF framework specified in this paper.

III. SIGN DESCRIPTION FRAMEWORK

Based on the quaternary model of sign, this section defines relevant concepts of SDF framework.

A. Bi-Type Tree

A *bi-type tree* is a tree that has two distinctive types of edge from the root vertex to children vertices of the root.

B. SDF Tree Data Model

SDF construction in expression is a collection of bi-type trees, represented by a tree model shown in Figure 2. In this specification, a bi-type tree has a root vertex called **identifier** which has two types of edges called **denotation edge** and **connotation edge**. The cardinality of denotation edge type is one. A denotation edge connects to a vertex called **denoter**. The edge of denoter is called **reification edge**, which connects to a vertex called **reifier**. The cardinality of connotation edge type is N. Each connotation edge connects to an identifier vertex i ($i \in$ N) of another bi-type tree. Connotation edge leads to the growth of a bi-type tree.

The assertion of a bi-type tree declares a sign. A collection of bi-type trees is called an **SDF tree**.

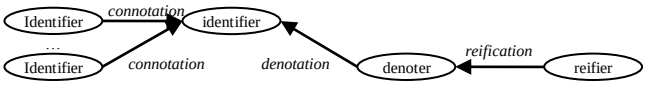

Figure 2. An SDF Tree Data Model

SDF tree data model consists of only edges of denotation, reification and connotation, and vertices of identifier, denoter and reifier. Both edges and vertices are objects in SDF specification.

C. Object

An **object** is a computable image of any object in reality such as an object of real, fictitious, physical, virtual, abstract, concrete, simple or complex.

There are two sets of objects, which are pairwise-disjoint. They are:

- literal
- sign

Any SDF computing systems are represented in signs and/or literals. Nevertheless, literals are only used in a limited scope to express any objects that are meaningless for computers to understand by default while a sign always meaningful.

D. Literal

A **literal** is an object that is not defined in SDF tree data model. It is assumed to be human-understandable but meaningless for computer to understand.

Literals are classified into two categories:

- *indefinitive literal*, including number such as "256" and "thirty seven", street address such as "Sunzhongshan Dajie", and serial number such as "iso76578" and "12568333444333".
- *linguistic literal*, including text, sentence, phrase, word and character.

Literals are for human to read, write and understand. For computer to read, write and understand, literals must be mapped onto signs, which are computer-readable, computer-writable and computer-understandable. This is because a literal can be associate with different concept to constitute different signs and cause ambiguous interpretation in meaning. For example, the literal "orange" can be a kind of fruits or a kind of color.

E. Sign

A **sign** is an object modeled by a **bi-type tree** defined in an SDF tree data model. Its semantics is defined by the following six components.

- an **identifier**
- a **denoter**
- a **reifier**
- a **denotation**, which is an arrowed relation from denoter to identifier, where the identifier of the arrowed side is the root vertex, and
- a **reification**, which is an arrowed relation from reifier to denoter, and
- zero or many **connotations**, which are arrowed relations from one identifier to another, where the identifier of the arrowed side is the root vertex.

1) Identifier

An **identifier** is a computer-readable and computer-understandable sign. It is used to uniquely convey the meaning of a sign created and used within SDF computing systems.

2) Denoter

A **denoter** is a sign where its meaning is readable and understandable by both computer and human being. It is used to describe the meaning of a sign. A denoter is equivalent to identifier in meaning such that the meaning of a denoter is exactly passed to an identifier.

Within a denoter, the meaning of a sign is created in two methods:

- **Method 1**: in a given *context*, an *interpreter* interprets an object (in reality) into the meaning of a sign about the object (in reality). The meaning, i.e. interpretation, is further transformed into a *concept*. The concept is expressed in a sequence of literals readable and understandable by human being.
- **Method 2**: the meaning of the sign from Method 1 is again transformed into a *structure*. The structure is readable and understandable by computer and is a sequence of identifiers of other signs.

Within a denoter, concept and structure are completely semantically equivalent in meaning as two aspects of a sign exactly like signifier and signified defined in Saussure' dyadic model. The difference here is that structure is computer-readable, computer-writable and computer-understandable while the concept is human-readable, human-writable and human-understandable.

A denoter reflects an abstract meaning of an object.

3) Denotation

A **denotation** is a relational sign. It conveys the concept of a denoter sign into the meaning of an identifier through the structure of a denoter.

A denotation guarantees that the meaning of an identifier is faithfully transformed from the meaning of a structure of a denoter to the meaning of an identifier.

4) Reifier

A **reifier** is a sign where its meaning is represented by a sequence of identifiers of other signs defined in an SDF dictionary and probably mixed with some indefinitive literals restricted by sign data types.

A reifier reflects a particular meaning of an object.

5) Refication

A **reification** is a relational sign. It reifies a denoter into a couple of (denoter, reifier).

A reification sign guarantees that a denoter changes from an abstract state into a reified state by combining a reifier into a denoter.

6) Connotation

A **connotation** is a relational sign. It introduces a child relationship by connoting an identifier as a parent sign through a set of other signs as children signs.

A connotation guarantees that an identifier as a sign has a set of fixed intensions again expressed as signs. It makes the meaning of an identifier as a sign to be richer. Thus, an identifier as a sign with connotation relationship is naturally becomes a compound sign, whose meaning is jointly expressed by the identifier and the connotation.

By definition, a sign is naturally recursive until only an identifier is left.

IV. TYPES OF SIGNS

In SDF, signs are typed for representing senses in different expressed forms. Three general types are categorized, which are basic sign type, standard sign type and degenerated sign type.

A. Basic Sign Types

Sign has four basic types, which are atomic sign, compound sign, abstract sign and reified sign.

1) Atomic Sign

Given a sign, when all connotations are blank, the sign is atomic and called an **atomic sign**.

2) Compound Sign

Given a sign, when there exists at least a connotation that is not blank, the sign is compound and called a **compound sign**.

3) Abstract Sign

Given a sign, when all reifications are blank, the sign is **abstract** and called an **abstract sign**.

4) Reified Sign

Given a sign, when there exists at least a reification that is not blank, the sign is **reified** and called a **reified sign**.

Table I illustrates the examples of atomic sign, compound sign, abstract sign and reified sign.

TABLE I. EXAMPLES OF BAISC SIGN TYPES

	Abstract Sign	**Reified Sign**
Atomic Sign	*Identifier:* 1111 (*Denotation:* (refrigerator, XYZ company))	*Identifier:* 1111 (*Denotation:* (refrigerator, XYZ company) *Reification:* (Haier 356))
Compound sign	*Identifier:* 1111 (*Denotation:* (refrigerator, XYZ company) *Connotation:* (*Identifier:* 2222 (*Denotation:* color) *Identifier:* 3333 (*Denotation:* door)))	*Identifier:* 1111 (*Denotation:* (refrigerator, XYZ company) *Reification:* (Haier 356) *Connotation:* (*Identifier:* 2222 (*Denotation:* color *Reification:* white) *Identifier:* 3333 (*Denotation:* door *Reification:* 2)))

Mapping onto the SDF tree data model shown in Figure II, basic signs can be summarized in Table 2.

TABLE II. STRUCTURE OF BASIC SIGN TYPES

	Abstract Sign	**Reified Sign**
Atomic Sign	identifier (*Denotation*(denoter))	identifier (*Denotation*(denoter) *Reification*(reifier))
Compound sign	identifier (*Denotation*(denoter) *Connotation*(identifier(*Denotation*(denoter)) identifier (*Denotation*(denoter))))	identifier (*Denotation*(denoter) *Reification*(reifier) *Connotation*(identifier(*Denotation*(denoter) *Reification*(reifier)) identifier(*Denotation*(denoter) *Reification*(reifier))))

In Table II, identifier is always a root vertex in a bi-type tree. A whole SDF tree is assembled through a connotation that connects the identifier of a parent bi-type tree and many identifiers of children bi-type trees.

B. Standard Sign Types

A **standard sign** is an abstract yet atomic sign where its denoter sign is disassembled into four smaller signs, such that:

standard sign ::= identifier(*Denotation*(**structure, concept, interpreter, context**))

where:

- **structure** is a sequence of identifiers of signs defined in an SDF dictionary. These identifiers are computer-readable, computer-writable and computer-understandable;
- **concept** is a sequence of literals not defined in SDF dictionary. Literals are human-readable, human-writable and human-understandable. They are often defined in dictionaries of various natural languages;
- **interpreter** is a person or a group of people interprets an object in reality into a sense or a meaning;
- **context** is an environment where an interpreter interprets any object in reality into a concept.

For these four signs, the meaning of the structure is equivalent to the concept and the concept is equivalent to the meaning interpretation by the interpreter at the context, such that:

structure ::= interpret(interpreter, context) =
interpretedSign(interpreterSign, contextSign), and
concept ::= interpret(interpreter, context) =
interpretedLiteral(interpreterLiteral, contextLiteral),

where interpret is an interpretation function taking interpreter and context as input and concept as output.

In this definition, the meaning of structure and the meaning of concept is equivalent by default. That is to say, interpreter is responsible for ensuring the meaning equivalence between structure and concept. The difference between structure and concept is that the former is computer-readable, computer-writable and computer-understandable and the latter is human-readable, human-writable and human-understandable.

The definitions of structure and concept here inform that the creation of a standard sign must consider the impacts of the intelligence of interpreter, influence of environment, and readability and understandability of both computer and human being. A question may be asked that how to ensure the meaning consistency across various enterprise systems. The approach we adopt is the introduction of a collaborative sign editing system, which is used as a common context for all involved sign designers to disambiguate the possible semantic conflicts between signs.

C. Degenerated Sign Type

Sign degeneration is a sign use method of replacing a standard sign with a simpler-formed sign by maintaining the same meaning between the simpler-formed sign and the standard sign. In this research, we define six types of degenerated sign types.

1) Peircian sign

A **Peircian sign** is a sign omitting the context sign from the denoter sign in a standard sign, such that:

Peircian Sign ::= identifier(*Denotation*(**structure, concept, interpreter**)),

where interpreter is responsible for generating the sense or meaning of the sign and transformed to concept and structure, and finally transformed to an identifier.

2) Community Sign

A **community sign** is a sign omitting the interpreter sign from the denoter sign in a standard sign, such that:

Community Sign ::= identifier(*Denotation*(**structure, concept, context**)),

where context is the environment in which a the sense or meaning of the sign is generated and transformed to concept and structure, and finally transformed to an identifier.

3) Saussurean Sign

A **Saussurean sign** is a sign omitting both interpreter sign and context sign from denoter in a standard sign, such that:

Saussurean Sign ::= identifier(*Denotation*(**structure, concept**)),

where concept is the sense or meaning of the sign for human use and is conveyed in structure for computer use, and finally transformed to an identifier.

4) Structure Sign

A **structure sign** is a sign omitting the signs of context, interpreter and concept from denoter sign in a standard sign, such that:

Structure Sign ::= identifier(*Denotation*(**structure**)),

where structure conveys the sense or meaning of a sign for computer use and finally transformed into an identifier.

5) Concept Sign

A **concept sign** is a sign omitting the signs of context, interpreter and structure from denoter sign in a standard sign, such that:

Concept Sign ::= identifier(*Denotation*(**concept**))

where concept is the sense or meaning and finally transformed to an identifier.

6) Symbolic Sign or Symbol

A **symbolic sign** or a **symbol** is a sign omitting the whole denotation of a standard sign, such that:

Symbolic Sign | Symbol ::= identifier

where the sense or meaning is simply conveyed in an identifier.

Symbol is the basis of all types of signs. A symbol is a sequence of binary codes in computer memory store, which is computer-readable and understandable. Each symbol maps onto a human-readable literal such as a character, a word, a phrase or a drawing.

Any symbol is implemented as a bit sequence, called **Sdfcode**. Its implementation is described in Sdfcode Binary Encoding Scheme, which will not be discussed in this paper.

All degenerated signs are equivalent to their original standard signs in the sense or meaning. The different form is only for different use purpose. However, any degeneration shall not cause semantically inconsistency problem.

Degenerated signs together with atomic, compound, abstract and reified signs are used to construct various SDF applications such as dictionaries, business documents, business processes, and logical expressions. For example, an SDF dictionary can be defined in a scheme of dictionary(term(id, structure, concept, context), interpreter).

V. COMPARING WITH EXISTING RDF APPROACH

The primary purpose of SDF is to define any object in reality from simple to complex. A simple object is defined through the denotation relationship, such that:

$$\text{sign} : \text{simple object} \xrightarrow{\text{denotation}} \text{identifier}$$

while a complex object is defined through both the denotation and connotation relationships, such that:

$$\text{sign} : \text{complex object} \xrightarrow{\text{denotation} \wedge \text{connotation}} \text{identifier}$$

In these two definitions, denotation provides a denotative definition as a concept for the structure of identifier to constitute an atomic sign. To notate a complex object in reality, connotations are involved to provide connotative definitions to form a compound sign.

Comparing with the existing RDF (Resource Description Framework) approach [3] to representing information resources, SDF has several advantages in resource (i.e. any object in reality) representation.

(1) Any resource in SDF is self-defined without ambiguity through denotation. This is because any resource must be represented as a sign consisting of structure (i.e., unique identifier) denoted by a unique concept at a certain context. There exists a bilateral relationship such that if SDF Identifier then SDF Concept and if SDF Concept then SDF Identifier.

(2) Any resource can be described in more details flexibly as necessary through connotations. This is because after any resource is represented as an atomic sign, users can connote this sign as necessary to depict its complexity as necessary without losing its original meaning of the sign.

(3) Any resource in SDF is capable of representing heterogeneous resources as long as they share the same denotation (which is a cross-context feature). This is because the unique identifier of a sign as a structure can map onto any other structures without changing the meaning.

(4) Any abstract resource can be reified without ambiguity in the reification level. This is because any reification in SDF is sign-based with non-ambiguous meaning.

In Table III, we compare SDF with RDF in the aspects of resource ambiguity, resource description (or resource modeling), resource cross-context ability, and reification ambiguity.

TABLE III. COMPARING SDF WITH RDF

	RDF	SDF
Resource representation ambiguity	Resource is identified by unique IRL or literal. (1) IRL ← referent only; (2) literal is ambiguous or without definition.	Resource is identified by unique SDF Identifier. SDF Identifier ↔ SDF Concept guarantees non-ambiguity of meaning. Literal is only for human purpose, not involving in computation.

Resource description or modeling	Resource is described or modeled in details by classes and properties through RDF triple data model. It is rather difficult for general users.	Resource is described or modeled in details by connotations through SDF tree data model. It is very easy for general users simply by adding connoted signs.
Cross-context heterogeneous resource mapping	Difficult. Cross-domain ad hoc mapping such that: map(IRL:dom_1, IRL:dom_2, …, IRL:dom_n)	Easy. Cross-context collaborative mapping such that: map(SDF Identifier, local identifier), see [4]
Ambiguity of reified resources (or individuals)	Disambiguating individuals (or instances) is not provided	Disambiguating reified resources is provided by requiring all reified sources represented in signs (i.e., using SDF identifiers).

Table I shows some basic advantages over the existing RDF resource representation approach. There are more advantages such as statement representation, data structure and neutrality of natural languages. However, these are beyond the general introduction of SDF in this paper and will be discussed elsewhere later.

VI. CONCLUSION

Sign Description Framework (SDF) is an abstract computational model representing any object in reality as a sign, which is both human and computer readable and understandable. It is used to depict any information resource used in Internet including semantic web and Internet of things. It has advantages of non-ambiguity of resource representation, easy resource modeling, and cross-context heterogeneous resource mapping. It can be widely applied in constructing dictionary, documents, rules, activities and processes, and services suitable for both computer and human being to read, write, interpret and reasoning.

There are several contributions in this paper. First, it has proposed a novel framework for representing information resources. Second, it has devised a new method of understanding resources by both human being and computers with equivalent sense of meaning. Third, it has provided a clear way of disambiguating the represented resources including their instances. There is much work not discussed in this paper yet, which includes representing various types of statements, texts in natural language, documents for e-business, rules for reasoning, and services for applications. These important researches will be gradually presented in future.

ACKNOWLEDGMENT

This research is partially supported by University of Macau research grant no. MYRG069(Y1-L2)-FST12-GJZ.

REFERENCES

[1] M. Atencia and M. Schorlemmer, "I-SSA: Interaction-Situated Semantic Alignment," in OTM 2008, Part I, LNCS 5331, R. Meersman and Z. Tari, Eds. Berlin: Springer, 2008, pp. 445–455.

[2] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," Scientific American, May 2001.

[3] R. Cyganiak, D. Wood and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax", W3C Recommendation 25 February 2014, http://www.w3.org/TR/rdf11-concepts/.

[4] J. Guo, "Collaborative Conceptualization: Towards a Conceptual Foundation of Interoperable Electronic Product Catalogue System Design," Enterprise Information Systems, vol. 3, no. 1, 2009, pp. 59-94.

[5] J. Guo and M. S. Ho, "Semantics-Enriched Document Exchange," in Proc. of ACM symposium on document engineering (DocEng'10), ACM Press, pp. 239-242, doi: 10.1145/1860559.1860613.

[6] G. Xiao, J. Guo, L. Xu and Z. Gong, "User Interoperability With Heterogeneous IoT Devices Through Transformation," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, May 2014, pp. 1486-1496.

[7] N. Niu, L. Xu and Z. Bi, "Enterprise Information Systems Architecture - Analysis and Evaluation," IEEE Transactions on Industrial Informatics, vol. 9, no. 4, Nov. 2003, pp. 2147-2154.

[8] C. S. Peirce, "Peirce on Signs - Writings on Semiotic by Charles Sanders Peirce," J. Hoopes, ed. Chapel Hill and London: The University of North Carolina Press, 1991.

[9] F. Saussure, Course in General Linguistics, Illinois: Open Court Publishing Company, 1986.