Modelling Equivalent Definitions of Concepts

Daniele Porello

Institute of Cognitive Sciences and Technologies, CNR
daniele.porello@loa.istc.cnr.it

Abstract. We introduce the notions of syntactic synonymy and referential synonymy due to Moschovakis. Those notions are capable of accounting for fine-grained aspects of the meaning of linguistic expressions, by formalizing the Fregean distinction between *sense* and *denotation*. We integrate Moschovakis's theory with the theory of concepts developed in the foundational ontology DOLCE, in order to enable a formal treatment of equivalence between concepts.

1 Introduction

We place our analysis of concepts within the foundational ontology DOLCE [2]. Concepts are there intended to model classification of entities of a domain according to some relevant perspectives, in case the intensional aspect of the classification matters for knowledge representation. For example, in legal domains and in social ontology, concepts are fundamental for representing *roles* such as president, delegate, and student [2]. Although the extension of the role "student" provides the class of entities that can be classified as students, it is the intensional aspect of "student" that specifies the relevant information about the meaning of "student", i.e. the conditions that are necessary for classifying something as a student. Concepts are indeed intended to capture those intensional aspects. The motivation for using DOLCE is that it formalizes a rich theory of concepts that allows for relating conceptual information with other types of information concerning the entities of a given domain. In DOLCE, concepts are defined by means of *descriptions*, that are intended as semantic entities that are used to refer to concepts. Since descriptions are our way to access concepts, we are lead to focus on the meaning of the description that is associated to the concept. In Fregean terms, we are not only interested in the *denotation* of the definitions of concepts, but also in their *sense*, that is, in the way in which the denotation is given. Consider the following example. According to the Italian Constitution, the "President of the Italian republic" is also the "President of the Superior Judicial Court". Although the two descriptions refer to the same individual, we want to say that they are referring to different concepts *because* they express distinct senses.

We introduce a language for expressing definitions of concepts and we discuss a number of equivalence relations between definitions. Our approach is based on the calculus of meaning and synonymy developed by Moschovakis [6, 1]. This calculus provides a formal interpretation of the crucial distinction made by Frege between sense and denotation. The calculus is based on a typed system for expressing the meaning of natural language expressions, in the style of the Montague grammar [3, 4]. We will integrate the calculus of meaning and synonymy with the treatment of concepts in DOLCE.

We shall see how to deploy the calculus of meaning and synonymy to decide whether two descriptions of concepts are equivalent, i.e. they define the same concept. Since the calculus is based on type theory, it is not embeddable in first order logic, therefore, we view it as an external module that interacts with DOLCE, rather than a part of the ontology. This is motivated by the fact that a first-order theory such as DOLCE is not sufficiently expressive to talk about the meaning of first-order terms and formulas.

Possible applications of a formal theory of equivalence of concepts concern the possibility of comparing the intensional aspects of conceptualizations provided by two agents' theories. The remainder of this paper is organized as follows. The next section presents the formal grammar based on type theory and the background of the calculus of meaning and synonymy developed by Moschovakis. Section 3 discusses the application of the calculus of synonymy with DOLCE.

2 Formal background

We present the *typed calculus of acyclic recursion* $\mathcal{L}^{\lambda}_{ar}$ defined by Moschovakis. We refer to [7, 6, 5, 1] for the details. The idea of a typed system to model natural language semantics can be briefly summarized as follows. The elements of a vocabulary (e.g. words in natural language) are associated to terms of a certain type. The type encodes the properties that are required for composing complex meanings, while the term specifies the semantic contribution of the element of the vocabulary. For instance, the word "red" is associated to the type of functions from entities to truth values. The term associated to "red" is the specific function that associates "true" to red entities and false otherwise.

As in Montague grammar, types are defined recursively from the basic types e for entities and t for truth values: TYPES $::= e \mid t \mid (\tau \to \tau')$

We assume a finite set of constants K, the vocabulary of our semantic grammar. The typing relation: $c : \tau$ indicates that c has type τ. Given a choice of K, the set of terms of the language of acyclic recursion $L^{\lambda}_{ar}(K)$ is defined as follows. For each type τ, we assume two distinct infinite sets of *variables*: *pure variable*: v^{τ}_0, v^{τ}_1, ..., and *recursion variable* or *locations*: p^{τ}_0, p^{τ}_1, ... Pure variables range over the domains of their types, whereas location variables are only assigned to terms.

The main innovation in Moschovakis's calculus is the *acyclic recursion* construction. Given a sequence of location variables, $p_1, \ldots, p_n$ and a set of terms $A_1, \ldots, A_n$, a *system of assignments* is an expression $\{p_1 := A_1, \ldots, p_n := A_n\}$ that means that a term A_i is assigned to the position p_i. The acyclic recursion is a term written as follows:

$$A_0 \text{ WHERE } \{p_1 := A_1, \ldots, p_n := A_n\}$$

The meaning of this statement is that, once one sets p_1 as A_1, ..., p_n is A_n, then one has A_0. The condition on the recursion construction is that its system of assignment must be *acyclic* [6].

Definition 1. *The set of terms of $L^{\lambda}_{ar}(K)$ is defined as follows:*

$$\text{TERMS} := c \mid v \mid p \mid B(C) \mid \lambda(v)(B) \mid A_0 \text{ WHERE}\{p_1 := A_1, \ldots, p_n := A_n\}$$

with the following conditions:

T1 *If c is a constant of type τ, then c is a term of type τ, $c : \tau$;*
T2 *Variables of type τ are terms of type τ, in particular $v : \tau$ and $p : \tau$;*
T3 *If $C : \tau$ and $B : \tau \to \tau'$, then $B(C) : \tau'$;*
T4 *If $B : \tau'$ and v is a pure variable of type τ, then $\lambda(v)(B) : \tau \to \tau'$;*
T5 *If for $n \geq 0$, $A_i : \tau_i$ and $p_1, ..., p_n$ are distinct locations $p_i : \tau_i$ such that the system of assignments $\{p_1 := A_1, \ldots, p_n := A_n\}$ is acyclic, then*

$$A_0 \text{ WHERE } \{p_1 := A_1, \ldots, p_n := A_n\} : \tau_0$$

For the sake of example, we treat a very simple fragment of English (Figure 1). We fix the set of constants K = { president, of, superior judicial council, italy, not, and, the, }[1] Next we associate semantic types to words.

Constants	Type
president	$e \to t$
of	$((e \to (e \to t)) \to (e \to t)$
superior judicial council	$e \to t$
italy	e
not	$t \to t$
and, or	$t \to (t \to t)$
the	$(e \to t) \to e$

Fig. 1. Grammar

By means of this simple grammar, we can compose terms to obtain complex terms and to check their types. For instance, we can compute that the string in natural language "the president of Italy" has type e, that is, it denotes an individual of type e. A fundamental preliminary step in order to move from natural language sentences to their correct semantic types and hence to their logical forms is to order the functional compositions in the correct way. In a number of approaches, this is done by means of a calculus that accounts for the syntactic structure of the sentence as providing the instructions to build the semantics [4]. For lack of space, we cannot enter the details here. For instance, we shall simply assume that it is possible to obtain the right order of applications from the order of words in natural language. Once we have the correct order of applications, we can compute the semantic of "the president of Italy" in a fully compositional way. In this case, functional applications (T3) is enough to obtain that the expression "the president of Italy" has to denote an element of e.[2] Consider now the term corresponding to "president of Italy", that is (of(italy)) (president). Its type is $e \to t$, since it denotes a class of entities. We can exemplify the acyclic constructor as follows. We allow also for possibly empty assignments, thus (of(italy)) (president) can also be written as follows:

$$\text{(of(italy))} \text{ (president)} \text{ WHERE } \{\ \} : e \to t$$
$$\text{(of}(p_1)\text{)} \text{ (president)} \text{ WHERE } \{p_1 := \text{italy}\} : e \to t$$

[1] For the sake of simplification, we treat "Superior Judicial Council" as single lexical entry.
[2] This step is called "semantic rendering" in [6].

$$(\mathsf{of}(p_1))\ (p_2)\ \text{WHERE}\ \{p_2 := \mathsf{president},\ p_1 := \mathsf{italy}\} : e \to t$$

2.1 The calculus of synonymy

Firstly, we introduce a syntactic notion of equivalence of terms. We say that A and B are *congruent*, $A \equiv_c B$, if and only if one can be obtained from the other by alphabetic change of bound variables (of both kinds) and re-ordering of the assignment within the acyclic recursion. For instance, $(\mathsf{of}(p_2))(p_1)$ WHERE $\{p_2 := \mathsf{president},\ p_1 := \mathsf{italy}\}$ is congruent with $(\mathsf{of}(p_3))(p_1)$ WHERE $\{p_3 := \mathsf{italy},\ p_1 := \mathsf{president}\}$.

Congruent terms are mere syntactic variants of one another, for that reason this is the strictest form of equivalence between terms. We are going to define the notions of *syntactical synonymity* and *referential synonymity*. The notion of syntactic synonymy is defined in [6] in terms of congruence of *canonical forms*. Canonical forms are defined by introducing a reduction calculus on terms which allows for computing effectively the canonical forms of terms [6, cf. par. 3.13]. The notion of canonical form provides the following definition of syntactic synonymity. Two constants will not be syntactically synonymous since they have non-equivalent canonical forms. For instance, "Italy" and the "Supreme Judicial Court" are not syntactically synonymous.

Definition 2 (Syntactic synonymity). *Two terms A and B are* syntactically synonymous*, $A \approx_s B$ if and only if their canonical forms are equivalent:*

$$A \approx_s B \Leftrightarrow cf(A) \equiv_c cf(B)$$

Example 1 (Canonical forms and syntactic synonymity). The canonical form of the term (of(italy)) (president) is given by: $(\mathsf{of}(p_1))\ (p_2)$ WHERE $\{p_2 := \mathsf{president},\ p_1 := \mathsf{italy}\}$. The term that expresses the meaning of "president of the Supreme Judicial Council" is: (of(sjc)) (president). Its canonical form is: $(\mathsf{of}(p_1))\ (p_2)$ WHERE $\{p_2 := \mathsf{president},\ p_1 := \mathsf{sjc}\}$.

The two canonical forms are not congruent, simply because they contain distinct constant terms: italy and sjc. Hence, although the two descriptions refer to the same individuals, the two terms are *not* syntactically synonymous.

Syntactic synonymity actually does not account for equivalence of *meanings* of the components. We shall see how to cope with that by means of the notion of referential synonymy. Referential synonymy is based on the notion of referential intension of a term, which intuitively models the process that computes the denotation of a term in a given model. This view corresponds to the Fregean idea that the "sense" of an expression is a way to compute its denotation [8]. In the following definition, we write $den(A)(g)$ to indicate the denotation of the term A under the assignment g.

Definition 3. *A is referentially synonymous with B, $A \approx_r B$, if and only if: (1) There exist suitable terms A_0, B_0, ..., A_n, B_n such that:*

$$A \Rightarrow A_0\ \text{WHERE}\ \{p_1 := A_1, \ldots p_n := A_n\}$$
$$B \Rightarrow B_0\ \text{WHERE}\ \{p_1 := B_1, \ldots p_n := B_n\}$$

(2) $\models A_i = B_i$, that is, for all assignments g, $den(A_i)(g) = den(B_i)(g)$

Referential synonymy takes into account the meaning of the constants that occur in a term. For instance, take two individual constants a and b that have the same denotation: let g be an assignment to the variables, $den(a)(g) = den(b)(g)$ are *not* syntactically synonymous, since they have non-equivalent canonical forms, although they are referentially synonymous, since their denotations coincide.

Example 2 (Referential synonymy). We can see that "the president of Italy" and "the president of the Supreme Judicial Court", although denotationally equivalent, are not referentially synonymous.

The expressions are rendered by (of(italy))(president) and (of(sjc))(president), whose canonical forms are:

$$(\mathsf{of}(p_1))\ (p_2)\ \text{WHERE}\ \{p_2 := \mathsf{president},\ p_1 := \mathsf{italy}\}$$

$$(\mathsf{of}(p_1))\ (p_2)\ \text{WHERE}\ \{p_2 := \mathsf{president},\ p_1 := \mathsf{sjc}\}$$

.

The terms are not referentially synonymous because the denotation of italy and sjc are not the same. By contrast, suppose K contains unemployed and not-employed. If we treat them as two constants, their denotations coincide, thus they are indeed referentially synonymous. Actually, in order to view them as referentially synonymous, we need to classify "not employed" as a single lexical entry. This amounts to deciding, at the level of semantic rendering, for unemployed and not-employed, whether the associated functions compute their reference by a different computation or not.

3 Application to concepts

In DOLCE, concepts are defined by *descriptions*. For instance, different agents may use different descriptions of the same concept, or a concept may be introduced by means of a description at a certain time. By relating concepts and descriptions, the view of DOLCE is that concepts manifest a certain dependence on the agents that use them to classify entities. For that reason, it is important to discuss notions of equivalence between descriptions. We only sketch the application of the notion of referential synonymy to the theory of concepts of DOLCE. Following [2], besides assuming a category for concepts C, we assume a category of descriptions DS. The categories are related by means of the relation of *definition* $\mathrm{DF}(x, y)$: a description x defines a concept y.

We assume that every element x of the category DS is a *name* for a term t_x in a suitable fragment of the language of acyclic recursion. Moreover, we assume that (syntactically) distinct terms are associated to distinct descriptions. We introduce two binary relations between descriptions $\sim_r$ and $\sim_s$: they are relational constants in DOLCE that are intended to represent $\approx_r$ and $\approx_s$, respectively. Since $\approx_r$ and $\approx_s$ are not fist-order relations, we do not attempt to provide a definition within DOLCE. We shall view the calculus of meaning and synonymy as an external (decidable) module that allows for computing whether $\approx_r$ and $\approx_s$ hold. That is, we are putting an external constraint on the models $\mathcal{M}$ of DOLCE that forces $a \sim_r b$ and $a \sim_s b$ to hold in $\mathcal{M}$ iff $t_a \approx_r t_b$ and $t_a \approx_s t_b$ are computable in the calculus of meaning and synonymy.

We can now define when two descriptions define the same concept. Two descriptions defines the same concept if they are referentially synonymous (a1). Moreover, the same concept cannot be defined by two descriptions that are not referentially synonymous (a2).

a1 $x \sim_r y \wedge DF(x, v) \wedge DF(y, w) \rightarrow v = w$

a2 $DF(x, v) \wedge DF(y, w) \wedge v = w \rightarrow x \sim_r y$

Even if two terms are denotational equivalent, but not referentially synonymous, they may still define two different concepts. For instance, as a theorem we can infer that although (of(italy)) (president) and (of(sjc)) (president) are denotationally equivalent, they define different concepts. Suppose that d_1 is the name of (of(italy)) (president) and d_2 is the name of (of(sjc)) (president). That is, d_1 and d_2 are elements of the category DS. One can establish in the calculus of meaning and synonymy [6] that

$$\text{(of(italy)) (president)} \not\approx_r \text{(of(sjc)) (president)}$$

Thus, we have that $d_1 \not\approx_r d_2$, that implies for our external constraints, that $d_1 \not\sim_r d_2$. Then, if $DF(d_1, v)$ and $DF(d_2, w)$, by axiom 2, v and w must be distinct concepts.

By contrast, a term that represent the word *student* in English and a term that represent the word student in Italian, say student and studente can be shown to be referentially synonymous, although they are not syntactically synonymous.

Bibliography

[1] E. Kalyvianaki and Y. N. Moschovakis. Two aspects of situated meaning. In *Logics for Linguistic Structures*, pages 57–86. Mouton de Gruyter, 2008.

[2] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social roles and their descriptions. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-2004)*, pages 267–277, 2004.

[3] R. Montague. The proper treatment of quntification in ordinary english. *Formal Semantics: The Essential Readings*, pages 17–34, [1973]2008.

[4] R. Moot and C. Retoré. *The logic of categorial grammars: a deductive account of natural language syntax and semantics*, volume 6850. Springer, 2012.

[5] Y. N. Moschovakis. Sense and denotation as algorithm and value. In J. Oikkonen and J. Väänänen, editors, *Logic Colloquium '90: ASL Summer Meeting in Helsinki*, pages 210–249. Springer-Verlag, 1993.

[6] Y. N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89, 2006.

[7] Y. N. Moschovakis. A logic of meaning and synonymy, 2010. Course notes, ESSLLI, Copenhagen.

[8] C. Penco and D. Porello. Sense as proof. In *New Essays in Logic and Philosophy*. College Publications, London, 2010.