Deriving Valid Expressions from Ontology Definitions

Yannis Tzitzikas[1,2], Nicolas Spyratos[3], Panos Constantopoulos[1,2]

[1] *Department of Computer Science, University of Crete, Greece*
[2] *Institute of Computer Science, ICS-FORTH*
[3] *Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France*
Email : {tzitzik, panos}@ics.forth.gr, spyratos@lri.fr

Abstract. In this paper, we consider an ontology as a set of terms together with three binary relations on terms, called synonymy, subsumption and cross-reference. We present a model-theoretic interpretation of ontologies and we show that this approach although appropriate for deciding the soundness of an ontology, is not sufficient for providing a sound and complete inference procedure for checking the validity of expressions in an ontology. Therefore a different "proof-theoretic" approach which allows checking the validity of expressions is also presented.

1 Introduction

Research on ontologies is becoming increasingly widespread in the computer science community and its importance is being recognized in many diverse research fields and application areas, such as conceptual analysis, conceptual modeling, information retrieval, information integration, agent communication, semantic annotation (see [11] for a review). There are numerous definitions of what an ontology is, revolving around the basic idea that "an ontology is a consensual and formal specification of a vocabulary used to describe a specific domain" (see [10] for a review).

In this paper, we consider an ontology as a set of terms together with three binary relations on terms, called *synonymy*, *subsumption* and *cross-reference*. These ontologies resemble to the structure of those linguistic ontologies (such as thesauri) which can accept a clear semantic interpretation. They also resemble to the organizational structures employed by web site providers, in order to organize their contents and to provide browsing and retrieval services, e.g. the subject hierarchy of Yahoo!. We present a model-theoretic interpretation of ontologies inspired by the approach of [25], [26] which allows checking whether a linguistic ontology is appropriate for tasks requiring a semantic interpretation of ontologies. We show that the model-theoretic interpretation although appropriate for deciding the soundness of an ontology, is not sufficient for providing a sound and complete inference procedure for checking the validity of expressions in an ontology. Therefore a different "proof-theoretic" approach is also presented.

In section 2 we define our ontologies and in section 3 we discuss their implementation. In section 4 we discuss the soundness of an ontology and in section 5 we present the query expressions that we consider. In section 6 we investigate the model theoretic interpretation as an inference procedure, while in section 7 we present an alternative inference procedure. In section 8 we compare our approach with other knowledge representation and reasoning approaches, and finally, in section 9 we review related work and conclude the paper.

2 Ontologies

Intuitively, an *ontology* consists of a set of words, or *terms*, and a set of *relationships* between the terms. Each term describes some aspect of a set of objects of interest and the relationships between terms reflect corresponding relationships between the objects. The assignment of meaning to terms and the recording of relationships are the outcome of a formal process and enjoy the consensus of some community. We conceptualize the world as a set of objects, that is, we assume an arbitrary, but fixed, domain of discourse and a corresponding set of objects Obj. The only constraint that we impose on the set Obj is that it must be a denumerable set.

Def 2.1 A *terminology* is a finite set of words pertaining to the objects in a specific domain of discourse. The elements of a terminology are called *terms.* $\diamond$

For example, the following set is a terminology for describing the content of university courses:

$$T = \{\ mathematics, history, computer\ science, humanities, ...\}$$

Here the underlying set Obj is the set of course identifiers, such as M315, H213, CS265 and so on. The set of objects described by a term is the *interpretation* of that term.

Def 2.2 Given a terminology T, we call *interpretation* of T over Obj any function $I : T \rightarrow 2^{Obj}$. $\diamond$

Thus each term t denotes certain objects in Obj and its interpretation $I(t)$ is the set of objects to which the term t is correctly applied. In our discussion the set Obj will be usually understood from the context. So we shall often say simply "an interpretation" instead of "an interpretation over Obj". In the previous example, an interpretation of T by a given agent might assign to the term $mathematics$ all course identifiers beginning with M, to $history$ all course identifiers beginning with H, to $computer\ science$ all course identifiers beginning with CS, and so on. However, note that different agents may attach different interpretations to the same term. Strictly speaking, interpretation, as defined above assigns to a term denotational or extensional meaning ([30]).

As we explained earlier, an ontology comprises not only a terminology, i.e. a set of terms, but also relationships between those terms. For the purposes of this paper, we shall consider three kinds of relationships, as stated in the following definition:

Def 2.3 An *ontology* is a quadruple $A = (T, \sim, \prec, \cdot)$ where T is a terminology and $\sim, \prec, \cdot$ are binary relations over T such that:

"$\sim$" is reflexive, symmetric and transitive (i.e. an equivalence relation). Conceptually, "$\sim$" means "*synonym of*", e.g. $math \sim mathematics$, $computer\ science \sim informatics$.

"$\prec$" is irreflexive, transitive, and asymmetric. Conceptually, "$\prec$" means "*subsumed by*" or "*isA*" or "*covered by*", e.g. $math \prec sciences$, $canaries \prec birds$.

"$\cdot$" is reflexive and symmetric. Conceptually, "$\cdot$" means "*cross-reference*" or "*related to*" in the sense of non-disjoint interpretations e.g. $math \cdot computer\ science$, $computers \cdot electronics$, $pets \cdot parrots$.

$\diamond$

Figure 1.(A) shows graphically an ontology for describing the content of university courses, where terms are represented as nodes and term relationships as labeled edges. Observe that edges labeled by "$\prec$" are oriented (because "$\prec$" is asymmetric), while edges labeled by "$\sim$", or "$\cdot$" are not oriented (because the relations "$\sim$" and "$\cdot$" are symmetric). Many web catalogs, such as Yahoo!, employ an ontology in order to organize their contents (for more see [21]) and Figure 1.(B) shows graphically an ontology for describing the contents of web pages advertising electronic products.

Now, let $A = (T, \sim, \prec, \cdot)$ be an ontology and let I be an interpretation of T. Clearly, in order for I to make sense in the ontology, it must also reflect all relationships that exist between terms. This is precisely what is stated in the following definition of a model.

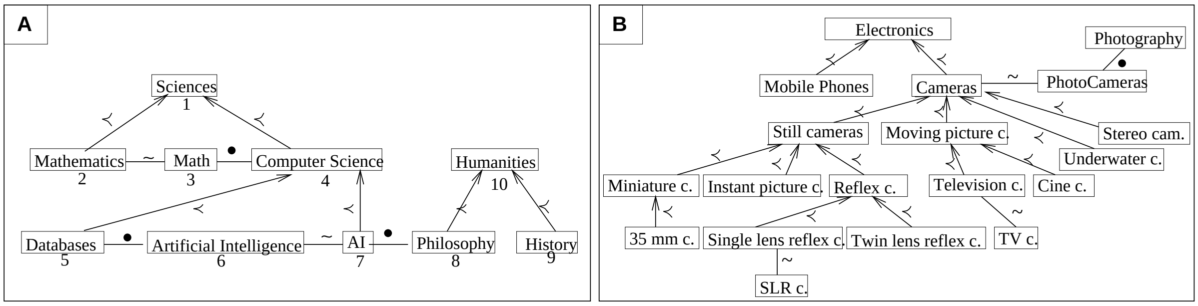


Figure 1: Graphical representation of two ontology definitions

Def 2.4 Let $A = (T, \sim, \prec, \cdot)$ be an ontology. An interpretation I of T is called a *model* of A if the following hold, for any terms t, t' in T:
(i) $I(t) \neq \emptyset$, i.e. every term is associated with at least one object (i.e. with a "witness")
(ii) if $t \sim t'$ then $I(t) = I(t')$, i.e. synonymy is interpreted as set equality
(iii) if $t \prec t'$ then $I(t) \subset I(t')$, i.e. subsumption is interpreted as strict set inclusion
(iv) if $t \cdot t'$ then $I(t) \cap I(t') \neq \emptyset$, i.e. term cross-reference is interpreted as nonempty intersection
$\diamond$

Let us now discuss these constraints. Recall that an interpretation, as defined in Def. 2.2 assigns to a term extensional meaning [1], and an ontology definition actually specifies a number of constraints that must hold between these meanings. However we must clarify that the interpretation of a term does not refer to its "extension" in a particular database, but it refers to its extension in the whole domain Obj, and this is the reason for requiring $I(t) \neq \emptyset$ for each term t and model I. On the contrary, $I(t) = \emptyset$ would mean that term t cannot by applied to any of the objects in Obj, but in this case, t would be useless and should not be included in the ontology definition. Another remark concerns the subsumption relation "$\prec$". We interpret subsumption by strict set inclusion ($\subset$) and not by set inclusion ($\subseteq$) as it is commonly done (see extensional subsumption [17]). Roughly, if subsumption was interpreted by set inclusion ($\subseteq$) then a "cycle" (cycles will be defined formally in section 4) might induce that a term t is synonym to a term t', although t may have been declared that subsumes t'. However this phenomenon does not fit well to the "axiomatic nature" of the recorded subsumption relationships of an ontology. Nevertheless, strict set inclusion ($\subset$) introduces problems when we want to extend a stored interpretation I to a model I' (which is needed in query answering in a materialized ontology). This issue is discussed in a subsequent article on materialized ontologies.

Def 2.5 An ontology A is called *sound* if there is a model of A, otherwise it is called *unsound*. $\diamond$

If A is a sound ontology we will write $A \models$, while if A is an unsound ontology we will write $A \not\models$. We will return to the issue of soundness in section 4.

3 Implementing an Ontology

The definition of an ontology can be implemented using any of a number of data models. For example, using the relational model [7], we can implement the definition of an ontology as a database schema consisting of four tables, one for storing the terminology and the others for storing the three relationships of the ontology:

TERMINOLOGY(term-id:`Int`, term-name:`Str`)
SYNONYM(term1:`Int`, term2:`Int`)
ISA(term1:`Int`, term2:`Int`)
RELATED(term1:`Int`, term2:`Int`)

[1]In contrast to the intensional meaning of terms (i.e. [14], [5]).

Note that each term of the terminology is stored in the form of a pair <term-id, term-name> where "term-id" is an internal identifier. For the purposes of this paper, however, we assume that term identifiers are integers and term names are strings. Each instance of this schema is called an *ontology base*. The rows of the tables shown of Figure 3 that are written in boldface correspond to the ontology base of the ontology shown in Figure 1.(A).

We call an ontology base *consistent* if it can be completed (using the properties of relations from Def. 2.3) so as to satisfy Def. 2.3. The tables of Figure 3 including the rows in italics constitute the completion of our ontology base. Note that two different ontology bases may have the same completion. However an ontology definition may be inconsistent, i.e. the ontology of Figure 2.(I) is inconsistent since it violates the irreflexiveness of "$\prec$", while the ontology of Figure 2.(II) is inconsistent since it violates the asymmetry of "$\prec$". Finally the ontology of Figure 2.(III) is inconsistent since the transitive closure of "$\prec$" violates the assymetry (and the irreflexiveness) of "$\prec$".

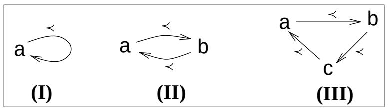


Figure 2: Three inconsistent ontologies

TERMINOLOGY

term-id	*term-name*
1	**Sciences**
2	**Mathematics**
3	**Math**
4	**Computer Science**
5	**Databases**
6	**Artificial Intelligence**
7	**AI**
8	**Philosophy**
9	**History**
10	**Humanities**

SYNONYM

term1	*term2*
2	**3**
6	**7**
3	*2*
7	*6*
1	*1*
2	*2*
3	*3*
4	*4*
5	*5*
6	*6*
7	*7*
8	*8*
9	*9*
10	*10*

ISA

term1	*term2*
2	**1**
4	**1**
5	**4**
7	**4**
8	**10**
9	**10**
5	*1*
7	*1*

RELATED

term1	*term2*
3	**4**
5	**6**
7	**8**
4	*3*
6	*5*
8	*7*

Figure 3: An ontology definition as a relational schema

4 Sound and Unsound Ontologies

The definition of an ontology for a given application domain involves collecting terms that are appropriate for that domain and reaching an agreement as to the relationships that hold among the terms. It is therefore natural to expect that some ontologies may not be sound. For example Figure 4.(A) shows three unsound ontologies. The ontology of Figure 4.(A).(I) is unsound, because there is no interpretation I such that $I(a) \subset I(b)$ and $I(a) = I(b)$. Similarly, the ontology of Figure 4.(A).(II) is unsound, because there is no interpretation I such that $I(a) \subset I(b) \subset I(c) = I(a)$. Finally, the ontology of Figure 4.(A).(III) is unsound because there is no interpretation I such that $I(a) = I(b) \subset I(c) = I(d) \subset I(a)$. However note that these ontologies are consistent, that is, they can be completed so as to satisfy Def. 2.3. In fact, the basic reason why these three ontologies are unsound is the presence of cycles. Indeed, if we lump together the terms of Figure 4.(A) that are synonyms (as shown in Figure 4.(B)) then in each case we obtain a cycle. The notion of cycle plays a central role in our treatment of ontologies. Indeed, as we show next, an ontology is sound if and only if there is no cycle between the equivalence classes induced by "$\sim$".

Let $A = (T, \sim, \prec, \cdot)$ be an ontology and let $T/\sim$ be the set of equivalence classes induced by "$\sim$" over T (recall from Def. 2.3 that "$\sim$" is an equivalence relation). We can extend the relation

"$\prec$" over the set $T/\!\sim$ as follows: for all c, c' in $T/\!\sim$, $c \prec c'$ iff there is $t \in c$ and $t' \in c'$ such that $t \prec t'$. We shall use the same symbol for both "$\prec$" and its extension over $T/\!\sim$, as the distinction will be clear from the context.

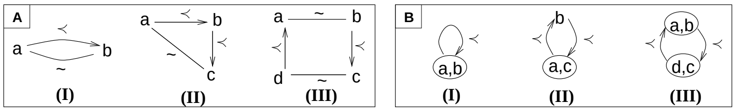


Figure 4: Three unsound ontologies

Def 4.1 An ontology $A = (T, \sim, \prec, \cdot)$ is *acyclic* if "$\prec$" is acyclic over $T/\!\sim$. Otherwise the ontology A is *cyclic.* $\diamond$

Recall that a binary relation R over a set N is acyclic, if for every element n_1 of N, there is no sequence $n_1, ..., n_k$ with $k \geq 1$ such that $n_1 = n_k$ and $n_i R n_{i+1}$ for all $i = 1, .., k-1$. For example, the ontologies of Figure 4.(A) are cyclic, because in each graph of Figure 4.(B) the set of nodes is the set $T/\!\sim$ and "$\prec$" is cyclic over $T/\!\sim$.

Proposition 4.1 Every cyclic ontology is unsound.
Proof:

Let $A = (T, \prec, \sim, \cdot)$ be a cyclic ontology, and assume that there exists a model I of A. It follows from Def. 2.4(ii) that all terms of an equivalence class $c \in T/\!\sim$ have the same interpretation, and let us denote this interpretation by $I(c)$. It follows from Def. 2.4(iii) that if $c \prec c'$ then $I(c) \subset I(c')$. Since A is cyclic there exists a sequence of equivalence classes $c_1, ..., c_k$ with $k \geq 1$ such that $c = c_1 \prec ... \prec c_k = c$. This implies that $I(c) = I(c_1) \subset ... \subset I(c_k) = I(c)$. Obviously, this is impossible, hence A is unsound. $\diamond$

Proposition 4.2 Every acyclic ontology A is sound. $\diamond$

We will prove this proposition by providing below an algorithm (Algorithm 4.1) which takes as input an acyclic ontology A and produces a model m of A. In this algorithm, we assume a function *witness* that takes as argument either a term t or a pair $t \cdot t'$ of different but related terms and returns a single object from Obj. We assume that *witness* is injective everywhere except on symmetric pairs, i.e. pairs of the form $t \cdot t'$ and $t' \cdot t$ with $t \neq t'$. For such pairs *witness* returns the same object, i.e. *witness*($t \cdot t'$) = *witness*($t' \cdot t$). Note that the existence of such a function requires that the set Obj is "adequately" large. Specifically, we must have: card(Obj) $\geq$ card(T) + 1/2 $*$ card($\{ t \cdot t' \mid t \neq t' \}$), where "card" stands for "cardinality". The correctness of the algorithm that follows, and therefore the proof of Proposition 4.2, pre-supposes satisfaction of the above constraint.

Let us apply the algorithm to the ontology shown in Figure 5, assuming Obj to be the set of all positive integers, and the function *witness* to be defined as follows:

$$witness(a) = 1 \quad witness(b) = 2 \quad witness(c) = 3 \quad witness(d) = 4$$
$$witness(e) = 5 \quad witness(f) = 6 \quad witness(b \cdot d) = 7$$

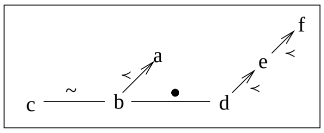
Below we describe each step of the algorithm, and Figure 6 shows the interpretations of the terms as they are being updated in each step of the algorithm.

Step 1: In this step a distinct integer is assigned to the interpretation of each term:
$m_1(a) = \{1\}, m_1(b) = \{2\}, m_1(c) = \{3\}, m_1(d) = \{4\}, m_1(e) = \{5\}, m_1(f) = \{6\}$
Step 2: In this step, the relationship $b \cdot d$ causes the following assignments:
$m_2(b) = m_1(b) \cup \{7\} = \{2\} \cup \{7\} = \{2, 7\}$
$m_2(d) = m_1(d) \cup \{7\} = \{4\} \cup \{7\} = \{4, 7\}$
Step (3): In this step, the relationship $c \sim b$ causes the following assignments:
$m_3(c) = m_2(c) \cup m_2(b) = \{3\} \cup \{2, 7\} = \{2, 3, 7\}$
$m_3(b) = m_3(c) = \{2, 3, 7\}$
Step (4): In this step, the relationships $b \prec a$, $e \prec f$ and $d \prec e$ cause the assignments:
$m_4(a) = m_3(a) \cup m_3(b) = \{1, 2, 3, 7\}$

Algorithm 4.1 Ontology Model
Input: An acyclic ontology $A = (T, \sim, \prec, \cdot)$
Ouptut: A model m of A

Step 1: For each t, set $m(t) := \{witness(t)\}$
Step 2: For each $t \cdot t'$
  If $m(t) \cap m(t') = \emptyset$ then
   $m(t) := m(t) \cup \{witness(t \cdot t')\}$
   $m(t') := m(t') \cup \{witness(t \cdot t')\}$
Step 3: For each $t \sim t'$
  If $m(t) \neq m(t')$ then
   $m(t) := m(t) \cup m(t')$
   $m(t') := m(t)$
Step 4: For each $t \prec t'$
  If $m(t) \not\subset m(t')$ then
   $m(t') := m(t) \cup m(t')$
Step 5: If changes in Step 3 or in Step 4 then
  Goto Step 3
 else return m

$m_4(f) = m_3(e) \cup m_3(f) = \{5, 6\}$
$m_4(e) = m_3(e) \cup m_3(d) = \{4, 5, 7\}$
Step (5): The control goes back to step 3.
Step (3): No changes happen in this step.
Step (4): In this step, the relationship $e \prec f$ causes the assignment:
$m_5(f) = m_4(e) \cup m_4(f) = \{4, 5, 6, 7\}$
Step (5): The control goes back to step 3 and since no other changes are done in steps (3) and (4), the algorithm terminates returning the produced model



ISA

b	a
e	f
d	e

SYNONYM

c	b

RELATED

b	d

Figure 5:

	Step 1	Step 2	Step 3	Step 4	Step 3	Step 4	**OUTPUT**
T	m_1	m_2	m_3	m_4		m_5	
a	{1}			{1,2,3,7}			**{1,2,3,7}**
b	{2}	{2,7}	{2,3,7}				**{2,3,7}**
c	{3}		{2,3,7}				**{2,3,7}**
d	{4}	{4,7}					**{4,7}**
e	{5}			{4,5,7}			**{4,5,7}**
f	{6}			{5,6}		{4,5,6,7}	**{4,5,6,7}**
Relationships that cause the assignments		$b \cdot d$	$c \sim b$	$b \prec a$		$e \prec f$	
				$e \prec f$			
				$d \prec e$			

Figure 6: Run of Algorithm 4.1 to the ontology base of Figure 5

Proposition 4.3 Given an acyclic ontology A, Algorithm 4.1 terminates and produces a model of A.
Proof:
Let m be the interpretation returned by Algorithm 4.1. According to Def. 2.4 the interpretation m is a model of A if for each t, t' of T:

(i) $m(t) \neq \emptyset$. This holds due to Step 1.

(ii) if $t \sim t'$ then $m(t) = m(t')$. This holds due to Step 3.

(iii) if $t \prec t'$ then $m(t) \subset m(t')$. This holds due to Step 4. If $t \prec t'$ the algorithm performs the assignment $m(t') := m(t') \cup m(t)$. This assignment guarantees only that $m(t) \subseteq m(t')$. However $m(t) \subset m(t')$ actually holds because $m(t)$ does not contain the element $witness(t')$ which belongs to $m(t')$ due to Step 1. If the element $witness(t')$ were also a member of $m(t)$ it would have been assigned to $m(t)$ only due to the existence of a sequence of equivalence classes $c_1 \prec ... \prec c_k$ ($k \geq 1$) such that $t' \in c_1$ and $t \in c_k$. But in that case, the relationship $t \prec t'$ and the above sequence would form a cycle which is a contradiction since the input ontology is acyclic.

(iv) if $t \cdot t'$ then $m(t) \cap m(t') \neq \emptyset$. This holds due to Step 2, and note that this property is preserved because the subsequent steps may only enlarge the interpretation of a term.

Clearly, Algorithm 4.1 terminates. Suppose it does not. Then, since the terminology T, and therefore $T/\sim$, is finite, this is possible only if the loop between Step 3 and Step 5 does not terminate which is possible only if there is a cycle in $T/\sim$. But this is impossible because the ontology is acyclic. $\diamond$

Theorem 4.1 An ontology A is sound iff it is acyclic.
Proof:
Follows immediately from propositions 4.1 and 4.2. $\diamond$

Two important remarks are in order here. The first is that Algorithm 4.1 produces a model of an acyclic ontology, even starting with an ontology base which might contain an "incomplete" ontology definition. The second remark is that by slightly modifying Algorithm 4.1 we can obtain an algorithm which takes as input any ontology A (cyclic or acyclic) and produces a model m of A, if A is sound, or returns "UNSOUND" if A is unsound. This is Algorithm 4.2 which in comparison with the Algorithm 4.1, has one extra if-then statement in Step 4, which is shown in boldface. It can be easily proved (similarly to the proof of Theorem 4.1) that an ontology is unsound iff the Algorithm 4.2 returns "UNSOUND". Figure 7 shows the application of this algorithm to the ontology of Figure 4.(III).

Algorithm 4.2 Ontology Model 2
Input: An ontology $A = (T, \sim, \prec, \cdot)$
Ouptut: A model m of A, if A is sound, otherwise "UNSOUND"

⋮
Step 4: For each $t \prec t'$
If $m(t) \not\subset m(t')$ then
$m(t') := m(t) \cup m(t')$
If $m(t) \not\subset m(t')$ then
return "UNSOUND"
⋮

Returning back to the constraints of Def. 2.4, we can now say that if subsumption was interpreted by set inclusion ($\subseteq$) then all ontologies would be sound, even the cyclic ones. Allowing cyclic ontologies has the following consequence. If a subset $C = \{c_1, ..., c_k\} \subseteq T/\sim$ forms a cycle, then all terms which are elements of one equivalence class in C, are essentially synonyms, although they may have not been declared as synonyms.

5 Querying an Ontology

The relationships contained in the definition of an ontology can be seen as *expressions* that combine terms of the terminology using the connectors " $\sim$ "," $\prec$ "," $\cdot$ ". There is an infinite set

	Step 1	Step 2	Step 3	Step 4	Step 3	Step 4	**OUTPUT**
T	m_1	m_2	m_3	m_4		m_5	
a	{1}		{1,2}	{1,2,3,4}		UNSOUND	**UNSOUND**
b	{2}		{1,2}		{1,2,3,4}		
c	{3}		{3,4}	{1,2,3,4}			
d	{4}		{3,4}		{1,2,3,4}		
Relationships that cause the assignments		$a \sim b$ $d \sim c$	$d \prec a$ $b \prec c$	$a \sim b$ $d \sim c$	$d \prec a$	

Figure 7: Run of Algorithm 4.2 to the ontology base of Figure 4.(III)

of expressions that one can form in this way, and the relationships contained in the definition of the ontology can be seen as the given *valid* expressions of the ontology. New valid expressions can be *inferred* from given ones, and the purpose of this section is to define the inference mechanism for doing so.

For instance, in Figure 1 the expression $Math \prec Sciences$ is not member of the completed definition of the ontology. However, this expression can be characterized as valid, because one can easily see that for every model I holds $I(Maths) \subset I(Sciences)$. The same holds for the expressions $Mathematics \cdot ComputerScience$, and $ComputerScience \cdot Humanities$. On the other hand, the expression $Math \cdot AI$, cannot be characterized as valid, since in the model I of the ontology, which is given below, $I(Math) \cap I(AI) = \emptyset$ (actually, this is the model produced by Algorithm 4.1).

$I(Sciences)$	=	{1,2,3,4,5,6,7,11,12,13}	$I(Mathematics)$	=	{2,3,11}
$I(Math)$	=	{2,3,11}	$I(Computer\ Science)$	=	{4,5,6,7,11,12,13}
$I(Databases)$	=	{5,12}	$I(Artificial\ Intelligence)$	=	{6,7,12,13}
$I(AI)$	=	{6,7,12,13}	$I(Philosophy)$	=	{8,13}
$I(History)$	=	{9}	$I(Humanities)$	=	{8,9,10,13}

This will lead us to define validity of expressions with respect to an interpretation. Before doing so, however, we must specify precisely what an expression is.

Def 5.1 Let $A = (T, \sim, \prec, \cdot)$ be an ontology. An *expression* over A is a description, a strict inclusion, an inclusion or a synonymy, defined as follows (where t is a term):

description $d ::= \epsilon | t.d$, where ϵ is the empty description
strict inclusion $si ::= d \prec d'$, where d, d' are descriptions
inclusion $i ::= d \preceq d'$, where d, d' are descriptions
synonymy $s ::= d \sim d'$, where d, d' are descriptions

expression $e ::= d | si | i | s$

◇

Note that a description is either empty, or it has the form $t_1 \cdot ... \cdot t_n$, where $n \geq 1$ and the t_i's are terms; we shall call each t_i a *subterm* of d. Thus every term of T and every cross-reference $t \cdot t'$ of an ontology definition, is a description. Also note that the notion of expression generalizes the notion of term and the relations between terms. Some examples of expressions over the ontology definition of Figure 1 follow:

$Math \cdot AI$
$Math \prec Sciences$
$Databases \cdot AI \prec ComputerScience$
$Math \sim AI$
$Databases \cdot AI \cdot Sciences$
$Math \prec AI$
$Mathematics \cdot AI \sim Math \cdot ArtificialIntelligence$

Let $A = (T, \sim, \prec, \cdot)$ be an ontology. Any interpretation I of T can be extended to an interpretation $\hat{I}$ over descriptions as follows: for any description $d = t_1 \cdot t_2 \cdot ... \cdot t_k$ over A, we define $\hat{I}(d) = I(t_1) \cap I(t_2) \cap ... \cap I(t_k)$, if $d \neq \epsilon$, and $\hat{I}(d) = \emptyset$ otherwise. For simplicity we shall use the symbol I to denote both the interpretation and its extension over descriptions.

Def 5.2 Let $A = (T, \sim, \prec, \cdot)$ be an ontology. We define *validity* of an expression in an interpretation I of T as follows:

a description d is valid in I, denoted $I \models d$, if $I(d) \neq \emptyset$.
a strict inclusion $d \prec d'$ is valid in I, denoted $I \models d \prec d'$, if $I(d) \subset I(d')$.
an inclusion $d \preceq d'$ is valid in I, denoted $I \models d \preceq d'$, if $I(d) \subseteq I(d')$.
a synonymy $d \sim d'$ is valid in I, denoted $I \models d \sim d'$, if $I(d) = I(d')$.
$\diamond$

Def 5.3 An expression e is *valid in* A, denoted $A \models e$, if $I \models e$ for all models I of A. $\diamond$

It follows immediately from Def. 2.4, that the expressions contained in the definition of an ontology A are valid in A. The main problem that we solve in this paper is the following: given an ontology A and an expression e decide whether $A \models e$.

Given a model m of A, let $E(m)$ denote the set of all expressions over A which are valid in m, that is, $E(m) = \{ e \mid m \models e \}$. Let us now denote by $E(A)$ the set of all expressions which are valid in A and we shall call $E(A)$ the *closure* of A. According to Def. 5.3 this set consists of the expressions which are valid in every model of A, that is,

$$E(A) = \bigcap \{ E(m) \mid m \text{ is a model of } A\}$$

6 Investigating a Model-theoretic Method for Deriving Valid Expressions

In this section we investigate a model-theoretic method checking the validity of expressions. In particular we look for a special model of A, denoted by m_A, such that $E(m_A) = E(A)$, that is, $m_A \models e$ iff $A \models e$ for all e. Note that by definition it holds $E(A) \subseteq E(m)$ for any model m of A. This means that the model-theoretic approach as inference procedure is always *complete* (that is $E(A)$ cannot contain an expression which is not element of $E(m)$ for any model m). Thus the difficulty lies in the *soundness* of the approach, that is, in finding a specific model m_A and proving that $E(m_A) \subseteq E(A)$. For example, Figure 8.(A) shows graphically an ontology A and a model m of that ontology in which $m \models b \cdot c$, although obviously $A \not\models b \cdot c$.

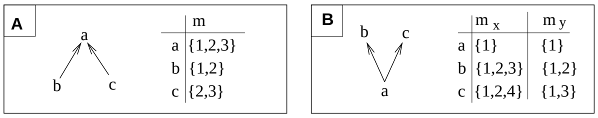


Figure 8:

Let us first investigate the descriptions. Let $E_d(m)$ denote the set of descriptions which are valid in a model m, and $E_d(A)$ denote the set of descriptions which are valid in every model of A. Now we will look for a model m such that $E_d(m) = E_d(A)$. Below we prove that all descriptions that are valid in m_{alg} (the model produced by Algorithm 4.1) are also valid in every model of A.

Proposition 6.1 $E_d(m_{alg}) = E_d(A)$
Proof: (see appendix A) $\diamond$

Thus for checking the validity of descriptions in A, we run once the Algorithm 4.1 which produces the model m_{alg}, and then it suffices to check the validity of descriptions in this model.

However the counter-example that follows proves that we cannot derive the validity of strict inclusions, or of synonymies, by the model-theoretic approach.

Example 6.1 Consider the ontology A shown in figure 8.(B). Clearly, in every model m of A, we have:

$$\left.\begin{array}{l} m(a) \subset m(b) \\ m(a) \subset m(c) \end{array}\right\} \Rightarrow m(a) \subseteq m(b) \cap m(c) \qquad (1)$$

This means that in some of the models of A, it holds $m(a) \subset m(b) \cap m(c)$ (see the model m_x), while in the rest of the models it holds $m(a) = m(b) \cap m(c)$ (see the model m_y). We conclude that the model-theoretic approach is not appropriate for checking the validity of strict inclusions and synonymies, since in the model m_A that we are looking for, it will either hold: $m_A(a) \subset m_A(b) \cap m_A(c)$, or $m_A(a) = m_A(b) \cap m_A(c)$, but none of these expressions is valid in every model of A. $\diamond$

Moreover we cannot check the validity of inclusions by the model-theoretic approach because if we could derive the validity of inclusions then we would be able to derive the validity of synonymies (since $A \models d \sim d' \;\Leftrightarrow\; A \models d \preceq d'$ and $A \models d' \preceq d$), but we have already showed that the latter is impossible in the model theoretic approach. Another counter example follows.

Example 6.2 Consider an ontology $A = (\{a,b\}, \emptyset, \emptyset, \{a \cdot b\})$. Clearly, in every model m, we have: $m(a) \neq \emptyset$, $m(b) \neq \emptyset$, and $m(a) \cap m(b) \neq \emptyset$. The top part of the table shown in Figure 9, presents four models, m_1, m_2, m_3, m_4, of A. These models demonstrate all possible relations that can hold between the interpretations of the terms a and b. In the bottom part of the table, each row corresponds to an expression over A, while each column corresponds to one of the four models. The presence of a bullet indicates that the corresponding expression is valid in the corresponding model. Observe that in each model, there is at least one expression, which is not valid in some or all of the other models. One can easily see that it is impossible to find a model m_A such that $E(m_A) = E(A)$ (always it will be $E(m) \supset E(A)$). $\diamond$

	overlapping	equal	subset	subset
	m_1	m_2	m_3	m_4
a	{1,3}	{1}	{1}	{1,2}
b	{2,3}	{1}	{1,2}	{1}
	$E(m_1)$	$E(m_2)$	$E(m_3)$	$E(m_4)$
$a \cdot b$	•	•	•	•
$a \sim b$		•		
$a \prec b$			•	
$b \prec a$				•
$a \cdot b \prec a$	•			•
$a \cdot b \prec b$	•		•	

Figure 9:

Thus in this section we proved that model-theoretic approach is a sound and complete inference procedure for checking the validity of descriptions. However it is unsound for inferring synonymies, inclusions and strict inclusions.

7 An Alternative Approach for Checking Expressions

In the previous section we showed that the model theoretic approach is not appropriate for deriving the validity of synonymies, inclusions, or strict inclusions. In this section we provide an alternative inference procedure for synonymies and inclusions, which is sound and complete.

Let $A = (T, \prec, \sim, \cdot)$ be an ontology and let $c_1, ..., c_n$ denote the elements of $T/\sim$. For each term $t \in T$ we denote by $c(t)$ the equivalence class of t.

Def 7.1 Given a description $d = t_1 \cdot ... \cdot t_k$, we call *reduction of d*, denoted by $r(d)$, the following subset of $T/\sim$: $\;r(t_1 \cdot ... \cdot t_k) \;=\; min_{\prec}\{\; c(t_1), ..., c(t_k)\;\}$ $\diamond$

This means that for a given description $d = t_1 \cdot ... \cdot t_k$, $r(d)$ is the subset of $T/\sim$ which contains the equivalence classes of the subterms of d (that is all $c(t_i)$), excluding those classes which are broader than other classes of the subterms in d [2]. For example, if t_i, t_j are subterms of d and if $c(t_i) \prec c(t_j)$ then the $c(t_j)$ is not an element of $r(d)$. Clearly if $r(d) = \{c_1, ...c_k\}$, then $c_i \not\prec c_j$ for all $i, j = 1..k$, and note that there is only one reduction for any given d.

Proposition 7.1 Let A be an ontology and d a description over A. For every model m of A it holds $m(d) = \bigcap_{c \in r(d)} m(c)$
Proof:

[2]This means that the set $r(d)$ is essentially the set of minimal elements of the set $\{c(t_1), ..., c(t_k)\}$.

Let $d = t_1 \cdot ... \cdot t_k$ and m be a model of A. It follows from Def. 2.4(ii) that all terms of an equivalence class $c \in T/\sim$ have the same interpretation, and let us denote this interpretation by $m(c)$. Thus $m(d) = m(t_1) \cap .. \cap m(t_k) = m(c(t_1)) \cap .. \cap m(c(t_k))$. Moreover, and according to Def. 2.4(iii), if $c \prec c'$ then $m(c) \subset m(c')$, thus $m(c) \cap m(c') = m(c)$. This implies that $m(c(t_1)) \cap .. \cap m(c(t_k)) = \bigcap_{c \in min_{\prec}\{c(t_1),...,c(t_k)\}} m(c)$. Hence, $m(d) = \bigcap_{c \in r(d)} m(c)$. $\diamond$

Lemma 7.1 Let d, d' be two descriptions over A. If $r(d) = r(d')$ then $m(d) = m(d')$ in every model m of A.
Proof:
According to Prop. 7.1, in every model m of A we have $m(d) = \cap_{c \in r(d)} m(c)$ and $m(d') = \cap_{c \in r(d')} m(c)$. Since $r(d) = r(d')$ we conclude that $m(d) = m(d')$. $\diamond$

Let d, d' be two descriptions over A. From the above lemma it is clear that if $r(d) = r(d')$ then certainly $A \models d \sim d'$. Thus we can use this method for checking the validity of synonymies. Moreover this method is complete, that is, $A \models d \sim d'$ iff $r(d) = r(d')$. This is proved by the next theorem.

Theorem 7.1 $A \models d \sim d' \Rightarrow r(d) = r(d')$
Proof: (see appendix A) $\diamond$

Concerning inclusions, we can reduce inclusion checking to synonymy checking:

Proposition 7.2 $A \models d \preceq d'$ iff $A \models d \cdot d' \sim d$
Proof:
($\Rightarrow$) $A \models d \preceq d'$ means that in every model m it holds $m(d) \subseteq m(d')$. The latter implies that $m(d) \cap m(d') = m(d)$, hence $A \models d \cdot d' \sim d$.
($\Leftarrow$) $A \models d \cdot d' \sim d$ means that in every model m it holds $m(d) \cap m(d') = m(d)$. The latter implies that $m(d) \subseteq m(d')$, hence $A \models d \preceq d'$. $\diamond$

Example 7.1 Consider the ontology shown in figure 8.(B), in which it is clear that $A \models a \preceq b \cdot c$, since in every model m, we have

$$\left.\begin{array}{l} m(a) \subset m(b) \\ m(a) \subset m(c) \end{array}\right\} \Rightarrow m(a) \subseteq m(b) \cap m(c)$$

According to our method $A \models a \preceq b \cdot c$ iff $A \models a \cdot b \cdot c \sim a$. The later is valid if $r(a \cdot b \cdot c) = r(a)$, which is true, since $r(a \cdot b \cdot c) = \{a\}$. Thus our method also derives that $A \models a \preceq b \cdot c$. Note that our method also derives that $A \not\models b \cdot c \preceq a$ since $A \models b \cdot c \preceq a \ \Leftrightarrow A \models a \cdot b \cdot c \sim b \cdot c$ which is false, since $r(a \cdot b \cdot c) = \{a\}$, while $r(b \cdot c) = \{b, c\}$. $\diamond$

Concerning strict inclusions, Figure 10 shows graphically some ontology definitions and some indicative strict inclusions which are valid (or invalid) in these ontologies. Checking the validity of strict inclusions is a more complex reasoning task and we shall present a sound and complete inference procedure which is based on the construction of a graph, on a subsequent paper. However in applications which employ ontologies in order to store descriptions of concrete objects, that is, in materialized ontologies we only need to check the validity of inclusions (i.e. for answering queries and checking the containment of queries).

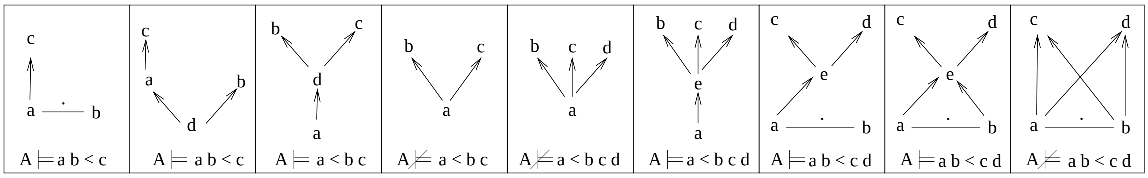


Figure 10: Examples of valid and invalid strict inclusions

8 Comparison with other Knowledge Representation and Reasoning Approaches

In this section we try to represent our ontologies in some logic-based languages, in particular, Propositional Calculus, First Order Predicate Calculus, Horn Clauses and Description Logics. We study each language in the following manner. At first we look for a method for translating an ontology definition A, to a set of well formed formulas Δ_A of that language and investigate whether the semantics and the corresponding inference rules of the language allow checking the soundness of A (that is, $A \models \ \overset{?}{\Leftrightarrow} \ \Delta_A \models$). If this holds, then we investigate whether an expression e (description, synonymy, inclusion, strict inclusion) can be written as a wff ϕ_e and whether the wff that are inferred from Δ_A correspond to expressions which logically follow from A and the vice versa (that is, $A \models e \ \overset{?}{\Leftrightarrow} \ \Delta_A \models \phi_e$). At last we discuss the inference procedures ($\vdash$) that are available for each language.

- **Propositional Calculus:** A candidate method for deriving a set of propositional wff is shown in the second column of Figure 11. Note that for each $\mathbf{t} \in T$, the set Δ_A contains an atom t, and clearly a model of Δ_A will satisfy each atom t. However this implies that each model of Δ_A will also satisfy all conjunctions of atoms, that is, all descriptions. From this we conclude that our ontologies cannot be represented in propositional calculus.

- **First Order Predicate Calculus:** Our ontologies can be represented in first order predicate calculus, and the third column of Figure 11 shows the method for producing Δ_A. Clearly an ontology A is sound iff the set of wffs Δ_A is satisfiable. Concerning expressions, the second column of Figure 12 shows how an expression e can be written as a wff ϕ_e, and clearly $A \models e \ \Leftrightarrow \ \Delta_A \models \phi_e$.

  Concerning the inference procedures of first order predicate calculus, we can check the satisfiability of Δ_A by applying resolution to the clause form of Δ_A, where reaching to an empty clause means that Δ_A is unsatisfiable, otherwise Δ_A is satisfiable. For checking the validity of a wff we can apply the resolution refutation [24] which is a sound and complete inference procedure for predicate calculus. However recall that predicate calculus is semi-decidable, and that even on problems for which resolution refutation terminates, the procedure is NP-hard - as is any sound and complete inference procedure for the first order predicate calculus [3].

- **Horn Clauses**: A candidate method for deriving a set of Horn Clauses [3] is shown in the forth column of Figure 11. For each term t, we produce a fact $t(C)$ where C denotes a new and unused constant [4]. For each relationship $t \cdot t'$, we produce two facts, $t(C), t'(C)$, where C denotes the same constant. However our ontologies cannot be represented with Horn clauses, because we cannot express the proper subset semantics of "$\prec$". Recall that the proper subset semantics of "$\prec$", are expressed in predicate calculus by the wff $\forall\ X\ [t(X) \rightarrow t'(X)]$ and the wff $\exists\ X\ [t'(X) \wedge \neg t(X)]$. The latter in clausal form becomes $t'(Sk) \wedge \neg t(Sk)$, where Sk is a new constant. For expressing this in Prolog would require two facts: $t'(Sk)$ and $\neg t(Sk)$ but the latter is not a Horn Clause. Thus we cannot express our ontologies with Horn Clauses

  Notice that if $t \prec t'$ was interpreted by $I(t) \subseteq I(t')$ then we would be able to represent our ontologies with Horn clauses. However our expressions (excluding descriptions) cannot be written as Horn goals, thus even in this interpretation of subsumption, we cannot check the validity of expressions with the inference procedures for Horn Clauses,

- **Description Logic:**
  Our ontologies can be represented in description logics (i.e. see [8]) and the last column of Figure 11 shows the method for producing a theory $\Delta_A = (\mathcal{T}_A, \mathcal{A}_A)$, where $\mathcal{T}_A$ is a *simple*-TBox, and the $\mathcal{A}_A$ is an empty ABox. Note that for each term t we do not produce any TBox statement, that is, we consider each term t as an *atomic concept* (a concept

[3] Horn clauses form the basis of the language Prolog.

[4] This resembles the way that resolution for predicate calculus eliminates existential quantifiers by Skolem functions. In particular the wff $\exists\ X\ t(X)$ becomes $t(Sk)$ where Sk is a new constant.

not appearing in the left side of any TBox-statement). For each $t \sim t'$ we derive two *primitive concept specifications* ($\dot{\leq}$), or a single *concept definition*, that is, a statement $t \doteq t'$. For capturing the proper subset semantics of "$\prec$" we need to define a new concept x as follows $x \dot{\leq} t' \sqcap \neg t$. Clearly, an ontology A is sound if the TBox of Δ_A is satisfiable,

Concerning expressions, Figure 12 shows how an expression e can be written as a DL expression ϕ_e. For checking the satisfiability of Δ_A we can use the *consistency* reasoning service of DL ($\Delta_A \not\models$). For checking descriptions we exploit the *consistency* reasoning service, in particular, a description d is valid if the TBox $\Delta_A \cup \{\neg\phi_d\}$ is unsatisfiable. For checking synonymies and inclusions we exploit the *subsumption check*. For checking strict inclusions we exploit both consistency and subsumption check. Concerning the inference procedures of DL tableaux calculus is a decision procedure for solving the problem of satisfiability.

A	Δ_A			
	Propositional Calc.	First Order Predic. Calc.	Horn Clauses	Description L.
t $\in$ **T**	t	$\exists X\ t(X)$	$t(C) :-$	
t $\sim$ **t'**	$t \rightarrow t'$ $t' \rightarrow t$	$\forall X\ [t(X) \rightarrow t'(X)]$ $\forall X\ [t'(X) \rightarrow t(X)]$	$t(X)\ :-\ t'(X).$ $t'(X)\ :-\ t(X).$	$t \dot{\leq} t'$ $t' \dot{\leq} t$
t $\prec$ **t'**	$t \rightarrow t'$	$\forall X\ [t(X) \rightarrow t'(X)]$ $\exists X\ [t'(X) \wedge \neg t(X)]$	$t'(X)\ :-\ t(X).$ $t'(C_t) :-$	$t \dot{\leq} t'$ $x \dot{\leq} t' \sqcap \neg t$
t $\cdot$ **t'**	$t \wedge t'$	$\exists X\ [t(X) \wedge t'(X)]$	$t(C) :-$ $t(C) :-$	$x \dot{\leq} t \sqcap t'$

Figure 11: Representation of our ontologies in some logic-based languages

e	ϕ_e	
	First Order Predicate Calculus.	Description Logic
d	$\exists X\ [t_1(X) \wedge ... \wedge t_k(X)]$	$t_1 \sqcap ... \sqcap t_k$
$d \sim d'$	$\forall X\ [(t_1(X) \wedge ... \wedge t_k(X)) \ \rightarrow\ (p_1(X) \wedge ... \wedge p_m(X))] \wedge$ $\forall X\ [(p_1(X) \wedge ... \wedge p_m(X)) \ \rightarrow\ (t_1(X) \wedge ... \wedge t_k(X))]$	$t_1 \sqcap ... \sqcap t_k \ \sqsubseteq\ p_1 \sqcap ... \sqcap p_m$ and $p_1 \sqcap ... \sqcap p_m \ \sqsubseteq\ t_1 \sqcap ... \sqcap t_k$
$d \preceq d'$	$\forall X\ [(t_1(X) \wedge ... \wedge t_k(X)) \ \rightarrow\ (p_1(X) \wedge ... \wedge p_m(X))]$	$t_1 \sqcap ... \sqcap t_k \ \sqsubseteq\ p_1 \sqcap ... \sqcap p_m$ and
$d \prec d'$	$\forall X\ [(t_1(X) \wedge ... \wedge t_k(X)) \ \rightarrow\ (p_1(X) \wedge ... \wedge p_m(X))] \wedge$ $\exists X\ [(p_1(X) \wedge ... \wedge p_m(X)) \ \wedge \neg\ (t_1(X) \wedge ... \wedge t_k(X))]$	$t_1 \sqcap ... \sqcap t_k \ \sqsubseteq\ p_1 \sqcap ... \sqcap p_m$ and $p_1 \sqcap ... \sqcap p_m \sqcap \neg t_1 \sqcap ... \sqcap \neg t_k$

Figure 12: Representing expressions

Here d denotes a description $t_1 \cdot .. \cdot t_k$ and d' denotes a description $p_1 \cdot ... \cdot p_m$.

9 Related Work - Concluding Remarks

The term ontology was first introduced in philosophy, where an ontology is a systematic account of existence, that is, a system of categories for a certain vision of the world (e.g. the Aristotle's ontology). Subsequently, the term ontology was employed in the field of *language engineering* which focuses on applications such as building NL interfaces. The ontologies of this area are often called "linguistic ontologies". Roughly, they consist of terms and relationships between terms, where terms denote lexical entries (words), while relations are intended as lexical or semantic relations between terms [5]. Initially, linguistic ontologies were written only for

[5]For instance in WordNet, terms are grouped into equivalence classes (called synsets). Each synset is assigned to a lexical category i.e. noun, verb, adverb, adjective, and synsets are linked by hypernymy/hyponymy and antonymy relations. The former is the subsumption relation, while the latter links together opposite or mutually inverse terms such as tall/short, child/parent (unclear semantics).

human use, however the evolution of computers enabled their representation in a digital form. Examples of such ontologies include WordNet, Mikrokosmos [19], SENSUS [15], and many others. In recent years, linguistic ontologies are not only used for language engineering, but also for a number of applications in information systems. Information retrieval is one example. In this area linguistic ontologies often play the role of controlled indexing languages. Moreover they may be used for assisting the query formulation process, or for expanding queries with synonyms hyponyms and related terms in order to improve recall (i.e. [22], [20], [21], [13]). They are also exploited for automatic classification of documents to concepts ([21]).

However there are many new applications of linguistic ontologies (see [11] for a review), i.e. linguistic ontologies are employed for information integration, for querying information sources, for achieving reuse in conceptual modeling (eg [9], [28], [23]). In many of these applications, and in contrast to NL applications, terms commonly are used to denote classes of entities of the domain, and a rising problem is that lexical relations do not always reflect semantic relations between classes of entities of the world (for more see [12]). These applications demand a semantic interpretation of a linguistic ontology. Our work can be seen as contributing in this direction. Thesauri are a case of linguistic ontologies. Commonly a thesaurus consists of a set of terms structured through relations as specified in [27]. Thesauri are quite close to the ontologies that we consider in this paper. In particular: the "$\prec$"-relation corresponds to the BTG-relation if it is used to connect terms with same criteria of identity, for example $Parrots$ BTG $Birds$; the "$\sim$"-relation corresponds to the USE-relation, for example $Maths$ USE $Mathematics$; the "$\cdot$"-relation corresponds to the RT-relation. If we look at some RT-relationships of existing thesauri, e.g. $Birds$ RT $Ornithology$, $Pathogens$ RT $Diseases$, $Aircraft$ RT $Anti-AircraftWeapons$, we infer that the relation RT is actually the union of many conceptual relations. This means that we cannot assign a clear semantic interpretation to this relation, therefore the RT relation is defined in thesauri as a symmetric relation. Our "$\cdot$"-relation corresponds to the special case of RT which is used to relate terms with overlapping meaning (e.g. $Ships$ RT $Boats$), which is not transitive (e.g. $\{cognitivescience \cdot computerscience, computerscience \cdot electronics\} \not\vdash cognitivescience \cdot electronics$), since nonempty set intersection is not transitive. Also note that in thesauri, terms are distinguished to preferred and to non-preferred terms. Each non preferred term has to be related with a preferred term by a USE relationship, and only preferred terms can participate in BTG, BTP, BTI, RT relationships. In our approach each term can participate in any kind of relationships; this allows a more direct and natural modeling.

Nowadays, the term "ontology" is used to denote quite different things such as dictionaries (e.g. WordNet, Webster's), thesauri (e.g. UMLS [6], AAT [7]), database schemas (oo-hierarchies), XML/RDF schemas [4], semantic networks, etc (have a look at the page [6]). In the area of knowledge representation, management and transfer, the most cited definition of an ontology is that of T. Gruber [9]: "*An ontology is an explicit specification of a conceptualization*" and examples of ontologies for structuring and reusing large bodies of knowledge, include CYC [16], KIF/Ontolingua [1], KA^2 [2]. In the web, the term ontology commonly refers to the content-based organizational structures employed by site providers, in order to organize their contents and to provide browsing and retrieval services, e.g. the Yahoo!'s subject hierarchy. Recently, more structured ontologies are employed for meta-tagging ([18], [29]). However note that the structuring of many available ontologies resembles to the structure of the ontologies presented in this paper. Also note that our ontologies are quite simple thus they can be understood, generated and maintained by average users. Note that the "bookmarks" facility of web navigators, i.e. Netscape, can be considered as a tool for defining "personal" ontologies

Summarizing our work we can say that in this paper we studied ontologies which consist of a terminology and three binary relations on terms, called synonymy, subsumption and cross-reference. For these ontologies we provided a model-theoretic interpretation and an inference mechanism for reasoning about their soundness and for deriving the expressions that are valid in an ontology. These issues are essential in almost any application of ontologies. Our future research concerns the indexing and retrieval of objects in a materialized ontology and

[6]Unified Medical Language System

[7]Arts & Architecture Thesaurus

the communication of ontology-based agents.

References

[1] ‘‘Knowledge Interchange Format”. Draft proposed American National Standard (dpANS) NCITS.T2/98-004, (http://logic.stanford.edu/kif/dpans.html).

[2] V. Benjamins, D. Fensel, and A. Prez. ‘‘Knowledge Management through Ontologies”. In *PAKM-98*, 1998.

[3] E. Borger. *‘‘Computability, Complexity, Logic”.* Amsterdam: North-Holland, 1989.

[4] Dan Brickley and R. V. Guha. ‘‘Resource Description Framework (RDF) Schema specification: Proposed Recommendation, W3C”, March 1999. http://www.w3.org/TR/1999/PR-rdf-schema-19990303.

[5] Mario Bunge. *Scientific Research I. The Search for Systems*, 3(1), 1967.

[6] Peter Clark. ‘‘Some Ongoing KBS/Ontology Projects and Groups”, 2000. http://www.cs.utexas.edu/users/mfkb/related.html.

[7] E. F. Codd. ‘‘A Relational Model of Data for Large Shared Data Banks”. *Communications of the ACM*, 13(6):377--387, 1970.

[8] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. *‘‘Reasoning in Description Logics”*, chapter 1. CSLI Publications, 1997.

[9] Thomas R. Gruber. ‘‘Towards Principles for the Design of Ontologies Used for Knowledge Sharing ”. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1994.

[10] Nicola Guarino. ‘‘Understanding, Building and Using Ontologies: A Commentary to ”Using Explicit Ontologies in KBS Development””. *International Journal of Human and Computer Studies*, 46:293--310, 1997.

[11] Nicola Guarino. ‘‘Formal Ontology and Information Systems”. In *Proceedings of FOIS’98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[12] Nicola Guarino. ‘‘Some Ontological Principles for Designing Upper Level Lexical Resources”. In *Proceedings of first int. Conf. on Language Resources and Evaluation*, Granada, Spain, May 1998.

[13] Nicola Guarino, Claudio Masolo, and Guido Vetere. ‘‘OntoSeek: Content-based Access to the Web”. *IEEE Inteligent Systems*, pages 70--80, May,June 1999.

[14] Raili Kauppi. ‘‘Einfuhrung in die Theorie der Begriffssysteme”. *Acta Universitatis Tamperensis, Ser A, Vol 15, University of Tampere*, 1967.

[15] K. Knight and S. K. Luk. ‘‘Building a Large-Scale Knowledge Base for Medicine Translation”. In *Proceedings of the 12th National Conf. on AI (AAAI’94)*, Seattle, Washington, 1994.

[16] D. B. Lenat. ‘‘CYC: A Large-Scale Investment in Knowledge Infrastructure”. *Communications of the ACM*, 38(11), 1995.

[17] H. Levesque and R. Brachman. ‘‘A Fundamental Tradeoff in Knowledge Representation and Reasoning (revised version)”. In *Readings in Knowledge Representation*. Morgan Kaufmann Publishers, 1985.

[18] Sean Luke, Lee Spector, David Rager, and Jim Hendler. ‘‘Ontology-based Web Agents”. In *Proceedings of First International Conference on Autonomous Agents*, 1997. (http://www.cs.umd.edu/projects/plus/SHOE/).

[19] K. Mahesh and S. Nirenburg. ‘‘A Situated Ontology for Practical NLP”. In *Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada, 1995.

[20] Zygmunt Mazur. ‘‘Models of a Distributed Information Retrieval System Based on Thesauri with Weights”. *Information Processing and Management*, 30(1):61--77, 1994.

[21] Deborah L. McGuinness. ‘‘Ontological Issues for Knowledge-Enhanced Search”. In *Proceedings of FOIS’98*, Trento, Italy, June 1998. Amsterdam, IOS Press.

[22] C. Paice. ‘‘A Thesaural Model of Information Retrieval”. *Information Processing and Management*, 27(5):433--447, 1991.

[23] B. Peterson, W.A. Andersen, and J. Engel. ‘‘Knowledge Bus: Generating Application-focused Databases from Large Ontologies”. In *Proceedings of the 5th Workshop KRDB-98*, Seattle, WA, USA, May 1998.

[24] J. Robinson. ‘‘A Machine-Oriented Logic Based on the Resolution Principle”. *Journal of the Association for Computing Machinery*, 12(1):23--41, 1965.

[25] Nicolas Spyratos. ‘‘The Partition Model: A Functional Approach”. Technical Report No. 430, INRIA, 1985.

[26] Nicolas Spyratos. ‘‘The Partition Model: A Deductive Database Model”. *ACM Transactions on Database Systems*, 12(1):1--37, 1987.

[27] International Organization For Standardization. ‘‘Documentation - Guidelines for the establishment and development of monolingual thesauri”, 1988. Ref. No ISO 2788-1986.

[28] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. ‘‘Towards Distributed Use of Large-Scale Ontologies”, 1996. USC/Information Sciences Institute, CA, September 27, 1996, (http://www.isi.edu/isd/banff_paper/ Banff_final_web/Banff_96_final_2.html).

[29] Frank van Harmelen and Dieter Fensel. ‘‘Practical Knowledge Representation for the Web”. In *Workshop on Intelligent Information Integration, IJCAI'99*, 1999.

[30] W. A. Woods. ‘‘Understanding Subsumption and Taxonomy”. In *Principles of Semantic Networks*, chapter 1. Morgan Kaufmann Publishers, 1991.

A Proofs of Propositions and Theorems

Proof of Proposition 6.1 :

First we introduce some notations. Given a term t, $Prod(t)$ denotes all descriptions which consist of one or more repetitions of t, i.e. $t, \ \ t \cdot t, \ \ ...$ etc. Clearly, if $d \in Prod(t)$ then $I(d) = I(t)$ in any interpretation I. Given a set of terms $S \subseteq T$, by $Prod(S)$ we denote all descriptions consisting of one or more repetitions of each term in S. For example $Prod(\{t, t'\})$ contains descriptions of the form: $t \cdot t', \ \ t \cdot t' \cdot t, \ \ t' \cdot t' \cdot t, ...$ etc. Clearly, if $S = \{t_1, ..., t_k\}$, and $d \in Prod(S)$, then $I(d) = I(t_1 \cdot ... \cdot t_k)$ in any interpretation I.

Below we prove that $E_d(m_{alg}) \subseteq E_d(A)$ by showing that the descriptions that are valid in m_{alg}, as it is procuded in each step of the algorithm, are valid in every model of A too.

Step 1: Let m_{S1} be the interpretation produced by Step 1 of the algorithm. Clearly a description d is valid in m_{S1} only if $d \in Prod(t)$ where $t \in T$. In any other case d is invalid: if d is the empty description then $m_{S1}(d) = \emptyset$, and if d contains two different subterms, then $m_{S1}(d) = \emptyset$ due to the injectivity of the function ”witness”. Thus we can write $E_d(m_{S1}) = \{Prod(t) | t \in T\}$.

Now let m be a model of A. Certainly $m(t) \neq \emptyset$ for all t in T, and also $m(d) \neq \emptyset$ for all $d \in Prod(t)$ for t in T. Hence we conclude that $E_d(m_{S1}) \subseteq E_d(A)$.

Step 2: Let m_{S2} be the interpretation produced by Step 2. Below we prove that $E_d(m_{S2}) \subseteq E_d(A)$.

Clearly, $E_d(m_{S2})$ includes the descriptions which are valid due to Step 1 (that is, all descriptions in $E_d(m_{S1})$). Below we prove by induction that $E_d(m_{S2})$ also includes the descriptions which are products of two related terms, that is, the descriptions $\{Prod(\{t, t'\})| \text{ where } \ t \cdot t'\}$. This means that we will prove that $E_d(m_{S2}) = E_d(m_{S1}) \cup \{Prod(\{t, t'\}) | t \cdot t'\}$.

Let $m_0, m_1, ...$ denote the interpretations produced by each iteration of the loop in Step 2, where m_0 is the interpretation produced by Step 1, that is $m_0 = m_{S1}$, and m_{j+1} is the interpretation produced by m_j after one iteration.

Let m_j be an interpretation in which $E_d(m_j)$ consists of descriptions of the form described above. Let m_{j+1} be the interpretation after one iteration of the loop in Step 2, and let d be a description such that $d \in E_d(m_{j+1})$, but $d \notin E_d(m_j)$. Let the difference between m_{j+1} and m_j be the following pair of statements:

$$\begin{aligned} m_{j+1}(t) &= m_j(t) \cup \{witness(t \cdot t')\} \\ m_{j+1}(t') &= m_j(t') \cup \{witness(t \cdot t')\} \end{aligned}$$

due to a relationship $t \cdot t'$ and the fact that $m_j(t) \cap m_j(t') = \emptyset$.

Certainly the description d should contain at least one (or both) of the terms t, t'. Let us assume that d contains only the term t, that is we may write $d = t \cdot x$ where x is a description which does not contain t or t'. We have:

$$m_{j+1}(d) \quad = \quad m_{j+1}(t) \cap m_j(x)$$

$$
\begin{aligned}
&= (m_j(t) \cup \{witness(t \cdot t')\}) \cap m_j(x) \\
&= (m_j(t) \cap m_j(x)) \cup (m_j(x) \cap \{witness(t \cdot t')\}) \\
&= m_j(d) \cup (m_j(x) \cap \{witness(t \cdot t')\}) \\
&= m_j(x) \cap \{witness(t \cdot t')\}
\end{aligned}
$$

Since $m_{j+1}(d) \neq \emptyset$, it must be $m_j(x) \cap \{witness(t \cdot t')\} \neq \emptyset$, that is, the object $witness(t \cdot t')$ must be an element of $m_j(x)$. But since the function "witness" is injective, the object $witness(t \cdot t')$ belongs only to $m_{j+1}(t)$ and $m_{j+1}(t')$. Thus, x should consist of repetitions of t or t', and x should not contain any other term. Thus the hypothesis that x does not contain the terms t or t' is false. Hence, we can write $E_d(m_{S2}) = E_d(m_{S1}) \cup \{Prod(\{t, t'\}) | t \cdot t'\}$.

Now let m be a model of A. If $t \cdot t'$ then certainly $m(t \cdot t') \neq \emptyset$, and clearly $m(d) \neq \emptyset$ for every $d \in Prod(\{t, t'\})$ since $m(d) = m(t \cdot t')$. Hence we conclude that $E_d(m_{S2}) \subseteq E_d(A)$.

Step 3: Let m_{S3} be the interpretation produced by Step 3. Below we prove that $E_d(m_{S3}) \subseteq E_d(A)$.

Let $m_0, m_1, ...$ denote the interpretations produced by each iteration of the loop in Step 3, where m_0 is interpretation produced by Step 2, that is $m_0 = m_{S2}$, and m_{j+1} is the interpretation produced by m_j after one iteration. We prove that $E_d(m_{S3}) \subseteq E_d(A)$ by induction, that is, we prove that if $E_d(m_j) \subseteq E(A)$ then $E_d(m_{j+1}) \subseteq E(A)$ too.

Let d be a description such that $d \in E_d(m_{j+1})$, but $d \notin E_d(m_j)$. Let the difference between m_{j+1} and m_j be the following pair of statements:

$$
\begin{aligned}
m_{j+1}(t) &= m_j(t) \cup m_j(t') \\
m_{j+1}(t') &= m_{j+1}(t)
\end{aligned}
$$

due to a relationship $t \sim t'$ and the fact that $m_j(t) \neq m_j(t')$.

Certainly the description d should contain at least one (or both) of the terms t, t'. Let us assume that d contains only the term t, that is we may write $d = t \cdot x$ where x is a description which does not contain t or t' We have:

$$
\begin{aligned}
m_{j+1}(d) &= m_{j+1}(t) \cap m_j(x) \\
&= (m_j(t) \cup m_j(t')) \cap m_j(x) \\
&= (m_j(t) \cap m_j(x)) \cup (m_j(t') \cap m_j(x)) \\
&= m_j(d) \cup (m_j(t') \cap m_j(x)) \\
&= m_j(t') \cap m_j(x)
\end{aligned}
$$

Let m be a model of A. Since $m_{j+1}(d) \neq \emptyset$, we have $m_j(t') \cap m_j(x) \neq \emptyset$ too. But since $E_d(m_j) \subseteq E(A)$, certainly it holds $m(t') \cap m(x) \neq \emptyset$. Due to $t \sim t'$ it must be $m(t) = m(t')$. Combining these two facts we obtain $m(t) \cap m(x) \neq \emptyset$, that is, $m(d) \neq \emptyset$. Hence we conclude $E_d(m_{S3}) \subseteq E_d(A)$.

Step 4: Below we prove that $E_d(m_{S4}) \subseteq E_d(A)$ where m_{S4} is the interpretation produced by Step 4.

Let $m_0, m_1, ...$ denote the interpretations produced by each iteration of the loop in Step 4, where m_0 is interpretation produced by Step 3, that is $m_0 = m_{S3}$, and m_{j+1} is the interpretation produced by m_j after one iteration. We prove that $E_d(m_{S4}) \subseteq E_d(A)$ by induction, that is, we prove that if $E_d(m_j) \subseteq E(A)$ then $E_d(m_{j+1}) \subseteq E(A)$.

Let d be a description such that $d \in E_d(m_{j+1})$, but $d \notin E_d(m_j)$. Let the difference between m_{j+1} and m_j be the following statement:

$$m_{j+1}(t') = m_j(t) \cup m_j(t')$$

due to a relationship $t \prec t'$ and the fact that $m_j(t) \not\subset m_j(t')$.

Certainly the description d should contain the term t', that is we can write $d = t' \cdot x$ where x is a description which does not contain t'. We have:

$$
\begin{aligned}
m_{j+1}(d) &= m_{j+1}(t') \cap m_j(x) \\
&= (m_j(t') \cup m_j(t)) \cap m_j(x) \\
&= (m_j(t') \cap m_j(x)) \cup (m_j(t) \cap m_j(x)) \\
&= m_j(d) \cup (m_j(t) \cap m_j(x)) \\
&= m_j(t) \cap m_j(x)
\end{aligned}
$$

Let m be a model of A. Since $m_{j+1}(d) \neq \emptyset$, we have $m_j(t) \cap m_j(x) \neq \emptyset$ too. But since $E_d(m_j) \subseteq E(A)$ this means that $m(t) \cap m(x) \neq \emptyset$. Due to $t \prec t'$ it must be $m(t) \subset m(t')$. Combining these two facts we obtain $m(t') \cap m(x) \neq \emptyset$. Hence $E_d(m_{S4}) \subseteq E_d(A)$.

Thus we proved that $E_d(m_{alg}) \subseteq E_d(A)$. Moreover, since m_{alg} is a model of A, we have $E_d(A) \subseteq E_d(m_{alg})$. Hence we conclude that $E_d(m_{alg}) = E_d(A)$. $\diamond$

Proof of Theorem 7.1

Let a, b be two descriptions over A, such that $r(a) \neq r(b)$. Assume that $r(a) = \{a_1, .., a_k\}$ and $r(b) = \{b_1, .., b_l\}$, where $k, l \geq 1$ Below we prove that we can always construct a model m_x, such that $m_x(a) \neq m_x(b)$. First note that since $r(a) \neq r(b)$, certainly it must be: (i) $r(a) \not\subset r(b)$, or (ii) $r(b) \not\subset r(a)$, or (iii) $r(a) \not\subset r(b)$ and $r(b) \not\subset r(a)$.

Assume case (i), that is, $r(a) \not\subset r(b)$. This means that there exists at least one a_x such that $a_x \notin r(b)$. Now consider a model m of A such that $m(a) = m(b)$. Below we describe a method for "enlarging" m, to a model m_x such that $m_x(a) \neq m_x(b)$. This method consists of the two steps:

Step 1. We add a new object o_b to each $m(b_i)$

Step 2. We add the object o_b, to the interpretation of each $c \in T/\!\sim$, such that $c \succ b_i$ for an $i = 1, .., l$.

Let m' denote the resulting interpretation after Step 1. Clearly, $o_b \in m'(b)$, while $o_b \notin m'(a)$, since o_b was not added to the interpretation of a_x. Note that the interpretation m' is not certainly a model of A. However, since m' was derived by enlarging a model m of A, certainly the constraints (i),(iv) of Def. 2.4, are also satisfied by m'. Moreover, since we are working on $T/\!\sim$, the constraint (ii) of Def. 2.4, is also satisfied by m'. However for satisfying the constraint (iii), we proceed to Step 2, and obviously the resulting interpretation is a model of A.

What remains to show is that Step 2 did not add the object o_b to each $m(a_i)$, since in that case it would be $m'(a) = m'(b)$. Clearly, this step would add o_b to each $m(a_i)$ only if:

$$\forall\ a_i\ \exists\ b_j\ :\ a_i \succeq b_j$$

Note that certainly it cannot be: $\forall\ a_i\ \exists\ b_j\ :\ a_i \sim b_j$ since in that case it would be $r(a) \subseteq r(b)$, and due to $r(a) \neq r(b)$, it would be $r(a) \subset r(b)$, which contradicts the hypothesis $r(a) \not\subset r(b)$. Thus, in view of our hypothesis, we can say that step 2 would update each $m'(a_i)$ only if:

$$\forall\ a_i\ \exists\ b_j\ :\ a_i \succeq b_j \text{ and } \exists\ a_x\ \exists\ b_x\ :\ a_x \succ b_x$$

From this formula we can see that it must be $b_x \in r(b) \setminus r(a)$, otherwise we would have $a_x \succ b_x = a_i$. Moreover notice that if $x = i$ then A would be cyclic (since we would have $a_x \succ a_x$), while if $x \neq i$ then this would contradict the definition of $r(a)$ (since it would be $a_x \succ a_i$. Therefore a_x should not be an element of $r(a)$). Thus we can rewrite the above formula as:

$$\forall\ a_x \in r(a) \setminus r(b) \quad \exists\ b_x \in r(b) \setminus r(a) \text{ such that } \quad a_x \succ b_x \tag{1}$$

Notice that this formula also implies that $r(b) \setminus r(a) \neq \emptyset$, that is $r(b) \not\subset r(a)$.

If formula (1) is satisfied then the enlargement process will result in a model m' such that $m'(a) = m'(b)$, which means that we failed to construct the model m_x that we are looking for. However in this case, we can try the opposite direction, that is, we can add a new object o_a to each $m(a_i)$. Certainly $o_a \in m(a)$ while $o_a \notin m(b)$ (since $r(b) \not\subset r(a)$). Now we enlarge m for making it a model. Again, what remains to show is that this step did not add o_a to each $m(b_i)$ too. By a similar analysis we reach the conclusion, that this step would update each $m(b_i)$ only if

$$\forall\ b_x \in r(b) \setminus r(a) \quad \exists\ a_x \in r(a) \setminus r(b) \text{ such that } \quad b_x \succ a_x \tag{2}$$

From this analysis we conclude that we cannot construct the model m_x that we are looking for, only if the formulas (1), and (2) both hold. Formula (1) imply that there exists $a_z \in r(a) \setminus r(b)$, and $b_z \in r(b) \setminus r(a)$, such that $a_z \succ b_z$. Formula (2) imply that there exists $a_{z'} \in r(a) \setminus r(b)$, such that $b_z \succ a_{z'}$. Notice that if $z = z'$ then A would be cyclic (since it would be $a_z \succ a_z$), while if $z \neq z'$ then this would contradict the definition of $r(a)$ (since it would be $a_z \succ a_{z'}$, therefore a_z should not be an element of $r(a)$). Thus we conclude that the formulas (1) and (2) cannot be both true.

This means that we can always construct a model m_x such that $m_x(a) \neq m_x(b)$. This implies that if $r(a) \neq r(b)$ then $A \not\models a \sim b$, that is, $A \models d \sim d' \Rightarrow r(d) = r(d')$. $\diamond$