MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 642 Revised, January 1982

Semantics of Inheritance and Attributions in the Description System Omega

Giuseppe Attardi and Maria Simi

Abstract

Omega is a description system for knowledge embedding which incorporates some of the attractive modes of expression in common sense reasoning such as descriptions, inheritance, quantification, negation, attributions and multiple viewpoints. A formalization of Omega is developed as a framework for investigations on the foundations of knowledge representation. As a logic, Omega achieves the goal of an intuitively sound and consistent theory of classes which permits unrestricted abstraction within a powerful logic system. *Description abstraction* is the construct provided in Omega corresponding to set abstraction. Attributions and inheritance are the basic mechanisms for knowledge structuring. To achieve flexibility and incrementality, the language allows descriptions with an arbitrary number of attributions, rather then predicates with a fixed number of arguments as in predicate logic. This requires a peculiar interpretation for *instance descriptions*, which in turn provides insights into the use and meaning of several kind of attributions. The formal treatement consists in presenting semantic models for Omega, deriving an axiomatization and establishing the consistency and completeness of the logic.

Keywords and Phrases: description, attribute, inheritance, knowledge representation, semantic network, logic, model, consistency.

CR Categories: 3.64, 5.21, 5.24.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support to the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract N00014-80-C-0505 and in part by the Office of Naval Research under Office of Naval Research contract N00014-75-C-0522.
This research was supported in part through a grant from Olivetti to the Artificial Intelligence Laboratory of MIT.
The second author has been supported in part by a fellowship of the Consiglio Nazionale delle Ricerche.

Table of Contents

1. Introduction

The need for formalisms for representing knowledge arises in several areas of computer science, for instance:

- problem solving and reasoning;

- program specification, synthesis and proving;

- natural language understanding.

Program specification is an area where the need for a large body of knowledge is immediately apparent. This is so because all the relevant knowledge has to be spelled out in complete detail and since most program domains are totally artificial, there is relatively little common or previous knowledge to build upon.

1.1. The Need for Structuring Knowledge

Predicate Logic is the most widely known formalism for expressing knowledge, but has also been extensively criticized either for being too general or too inflexible.

General deductive procedures for predicate logic easily go out of control. There have been a few attempts to design systems which include facilities for (loosely) controlling the reasoning process [8, 6, 14, 15]. They have provided significant insights in the control patterns used in deductions but very limited performance improvements.

Most of the systems based on predicate logic rely on the implicit assumption that the smartness of the system can be concentrated in its deductive capabilities. Research efforts have been devoted to devise algorithms that would avoid making irrelevant or redundant deductions and following non promising paths. Such kind of reasoning systems were expected to be able to reach the appropriate conclusions from the facts supplied, even though these facts were expressed in a totally disorganized way.

At the other side of the spectrum one could conceive a system where all valid facts are explicitly represented and organized for ease of retrieval. The reasoning process could then be based just on matching and retrieval capabilities. This suggests that the complexity of the deductive procedures might be traded for a richer, more structured base of knowledge with less sophisticated deductive capabilities.

Predicate logic provides little help in structuring knowledge. Its uniform notation does not allow to express which facts are related to each other.

1.2. The Need for a Logic Foundation

In recent years the lack of a satisfactory reasoning system has encouraged some experimentation with knowledge representation languages which require each deductive step to be explicitly user programmed. This enables different reasoning processes or heuristics to be devised or tailored to each domain [2, 3, 25].

Since the deductive knowledge inside these systems is procedurally defined, it is very difficult to understand how knowledge is used. Furthermore the possibility of building a system which examines itself is completely

out of question. Therefore while these systems are very useful as tools for experimentation or production, some interesting research issues are not addressed by them, like structuring and reorganization of knowledge, knowledge acquisition and introspection.

Some other knowledge representation systems embed deductive mechanisms of their own, most often procedurally defined [9, 25]. Such deductive mechanisms are not formally investigated, so their logical soundness can be questioned. It is in fact the case that some of these systems perform deductions which are not logically sound. Such is the case for Fahlman's NETL system [9], which is affected by the so-called "copy confusion" problem.

Fahlman's system has also both expressive and deductive limitations [23]. Its deductive capabilities are mainly oriented towards the associative retrieval of information, while the rest of the reasoning process must be performed outside the system. NETL most significant contribution is in showing that the network structure can be exploited by fast parallel search algorithms which perform marker propagation. Fahlman suggests that these algorithms could be embedded into a special purpose hardware architecture, built out of thousands of processing elements. The idea is appealing and we think that it can be developed even further. For instance, if a fully deductive mechanism can similarly be realized as the concurrent activities of a large number of very simple agents, then it would be possible to design a homogeneous architecture where a single type of component would provide memory capabilities and also perform reasoning activities. Once again, to design such a system, the basic rules of the reasoning process need to be carefully investigated.

A well understood logic formalism is still the most appropriate foundation for knowledge representation and reasoning. In a logic formalism, the basic deductive mechanisms are isolated from the intricacies of specific programming languages or implementations. Its power, its properties or its deficiencies can be more easily established. If a proof procedure is proposed for implementing the logic, results such as the correctness or the completeness of such algorithm can be stated or established only with respect to the theory.

We conclude that a knowledge representation formalism should:

- have a sound logic foundation
- provide mechanisms for useful modularization of knowledge, to help focusing the attention during reasoning
- have deductive rules which are simple, general, explicit, and accessible (to the system itself)

One approach to the structuring and modularization of knowledge is that which has evolved from the work on *semantic networks*. Knowledge is arranged in a network, where *nodes* represent concepts and *links* connect related concepts.

Omega embodies two ideas for structuring knowledge that are distilled from semantic networks: *inheritance* and *attributions*. Inheritance provides a way for grouping or partitioning knowledge. Attributions are used to build associative memory structures. The semantics provided in this paper fully characterizes these concepts and clarifies several issues related to the use of attributions, so that the semantic ambiguities and some of the

inadequacies of semantics networks [4] are resolved.

Omega is based on *descriptions*, which are a generalization of the notion of type as used in programming languages [26, 16, 12] and in program specification formalisms [5, 10]. Descriptions are also closely related to sets, indeed their semantics is given in term of sets of individuals.

Omega is a *calculus of descriptions* rather then a *calculus of predicates* as ordinary logic. The concept of description in logic, and in particular of *definite descriptions*, can be traced back to the work of Frege and Russell [21]. Logicians have always been bothered by the semantic problems raised by definite descriptions when none or more than one individual meets the description. Therefore they have favored their elimination by showing how descriptions can be contextually replaced by means of other constructs [22]. In Omega we deal with *indefinite descriptions* such as the description "(***an*** Integer)".

Omega is a type free system, in the sense that a single logical type is admitted, namely descriptions. Omega is an intuitively sound and consistent theory of classes which permits unrestricted abstraction within a powerful logic system. *Description abstraction* is a construct provided in Omega that is similar to set abstraction. Abstraction adds considerable expressive power to the language. Nevertheless, to avoid inconsistencies or paradoxes, the rules for abstraction had to be carefully designed. The proof of consistency that is provided in the paper is necessary to substantiate this claim.

Omega gains expressive power by providing the following features, some of which lack to other knowledge representation languages:

1. *variables*: this enables to express complex general relations, rather then isolated assertions as in most semantic network formalisms;

2. *quantification* over descriptions which encompasses quantification over both individuals and classes. This feature in itself produces a more expressive system then first order predicate logic;

3. *negation*: this removes the major limitation of systems like PROLOG, [8] based on the use of Horn clauses;

4. *abstraction*: allows description of classes of individuals in terms of their properties.

We expect that reasoning systems based on Omega would exploit parallel deduction strategies, possibly on some parallel hardware. Many simple deductions in Omega can be performed by searching through the knowledge network, where the span of the search can be circumscribed by the grouping of descriptions and by exploiting negative information. In general, the capabilities of parallel problem solving systems such as Ether [14] appear promising as a support for the reasoning mechanisms demanded by a rich system such as Omega. For instance the ability to process concurrently *proponents* and *skeptics* of the same goal enables sentences containing negations to be dealt with properly. Conversely, it is the provision of negative facts that makes the proponents/skeptics metaphor profitable and effective. So far the limitations on the use of negation seem to have been dictated mainly by considerations related to sequential proof algorithms.

The logic system of Omega is similar to that presented by Richard Martin in [17], even though we started with

different aims and motivations. Martin's system is proposed as a system of mathematical foundation, in alternative to the classical theory of sets. Martin does not present a proof of consistency for his system.

The present discussion of the semantics of Omega does not cover some other features of the system as metadescriptions, higher order capabilities, and the viewpoint mechanism.

An experimental system of active office forms has been implemented on the MIT LISP Machine by the authors in which Omega has been used to describe the underlying base of knowledge [1]. A second implementation of a reasoning system based on Omega is currently under development by Gerald R. Barber. A subset of Omega is being used to describe two dimensional objects within the SBA system by Peter de Jong [7].

In our experience the axiomatization of the logic has proven to be an extremely valuable guideline for developing an implementation.

1.3. Outline of the paper

A brief introduction to the logic theory Omega is presented in section 2. Section 3 describes the syntax of the language. In section 4 the models for the logic are presented and its semantics is defined. From this semantics, an axiomatization of the logic is derived in section 5. The completeness and consistency of the system is proved in sections 6 and 7. Section 8 is concerned with a discussion of attributions: the language is extended by introducing syntactic notations for four different kinds of attributions which are defined in terms of a single primitive kind of attribution. A number of properties are derived for each kind of attribution.

2. Descriptions and Predications

This section is an informal introduction to the language and the logic of Omega, to aid the reader in understanding section 4.

The simplest kind of description is the constant description, like:

Boston

or

3

Here the names Boston and 3 are names describing individual entities.

An *instance description* is a way to describe a collection of individuals. For instance

(*a* City)

or

(*an* Integer)

represent the collection of individuals in the class of cities or of integers.

The most elementary sentence in Omega is a predication. A predication relates a *subject* to a *predicate* by the

relation *is*. For instance the predication

Boston *is* (*a* City)

is understood to assert that the individual named Boston belongs to the class of cities.

Predication can be used to relate arbitrary descriptions. For instance the sentence:

(*a* Human) *is* (*a* Mortal)

states the fact that any individual of class human is also an individual of class mortal.

One of the fundamental properties of the relation ***is*** is *transitivity*, that allows for instance to conclude that

Socrates is (*a* Mortal)

from

Socrates *is* (*a* Human) and **(*a* Human) *is* (*a* Mortal)**

The description operators ***and***, ***or*** and ***not*** allow us to build more complex descriptions, like in the following examples:

(*a* Boolean) *is* (*true or false*)

(*a* Positive-Number) *is* (*not* (*a* Negative-Number))

Descriptions form a boolean lattice, determined by the partial ordering relation ***is***. The bottom of the lattice is the description ***Nothing***, a special constant playing the role of description of the null entity. The top of the lattice is the description ***Something***, another special constant which represents the less constrained, most generic description.

More complex statements can be built by combining statements with the logical connectives $\wedge$, $\vee$, $\neg$ and $\Rightarrow$, as in:

(Tom *is* ((*a* Cat) *or* (*a* Dog))) $\wedge$ $\neg$ (Tom *is* (*a* Dog)) $\Rightarrow$ Tom *is* (*a* Cat)

The difference between description operators and statement connectives is illustrated by the following examples:

(true* $\wedge$ *false) is false

(true and false) is Nothing

Attributions are attached to instance descriptions and serve the purpose of specializing a class by describing some of its properties. Alternatively, one can think of instance descriptions with attributes as parameterized descriptions.

Properties or attributes are always relative to a class. An individual may have some attributes when it is considered as an instance of a class and different attributes as instance of another class. It follows that attributes do not necessarily migrate (are not inherited) from one concept to another as is the case with many knowledge representation systems [9, 25, 18].

Attributions are a way of relating descriptions, as for instance in:

my-car *is* (*a* Car (*with* owner (*an* Italian)))

where "owner" is an attribute name for the concept "car", with attribute the description **(*an* Italian)**. The attribute named "owner" establishes a relationship between **my-car** and an Italian who is its owner. It is also a way to state the existence of some Italian who is the owner of **my-car** without explicit use of existential quantifiers.

Predicate logic expresses relationships of this kind by means of predicates. The relational data base model in computer science uses a similar concept of relations indexed by attribute names. Note that the semantics of both Predicate Logic and relational data bases depend on the fact that relations or predicates have a fixed number of arguments. In Omega attributes can be added or omitted from a description, with no *a priori* limit. By adding an attribution one gets a further specified description; omitting an attribution one obtains a more general description. So for instance:

(*a* Car (*with* owner (*an* Italian)) (*with* make VW))

is more specific than **(*a* Car (*with* owner (*an* Italian)))**. Attributions act like constraints that restrict the extension of a description.

The semantics that we will describe clarifies a number of problems connected with the use of attributions. In fact attributions have sometimes been used for expressing different incompatible ideas:

- binary relations (like whole/part relations);

- functional relations;

- general n-ary relations.

To illustrate some of the ambiguities in the use of attributions, consider the following example:

6 *is* (*a* Quotient (*of* dividend 12) (*of* divisor 2))

Here 12 is related to 6, the result of the quotient, and 2 also is related to 6. However, the values 12 and 2 are also related with one another as the arguments of a quotient. In fact, given the value of the quotient and either one of them, the other one is uniquely determined. Because of such dependency between 12 and 2, it does not seem appropriate to talk about 12 or 2 as attributes of the number 6 in the same sense as in the previous examples. In this case attributions are used to express relations among attributes themselves, besides expressing a relation between each of them and the whole description.

We use a different syntax to reflect the different nature of attributions. More precisely we use ***with*** when the attribute is solely related to the objects denoted by the description, as in:

(*an* Object (*with* color (red *or* green)) (*with* shape round))

(*a* Car (*with* passenger John) (*with* passenger Jane))

Instead we will use *of* when that is not the case. For example in:

(*a* Quotient (*of* dividend 12) (*of* divisor 2))

the attribute named "dividend" depends on the numbers that we are describing but also on the description of

the other attribute. The same is true for the "divisor" attribute. Another example is:

(*a* Segment (*of* origin 3.5) (*of* angle 45°) (*with* length 20))

In fact the same segment can be described in two ways by giving the origin of the segment and its direction (the angle between the line starting at the origin of the segment and the cartesian axes). Given the segment there are two possible origins and the angle depends on the the origin chosen to describe the segment. For example:

(*a* Segment (*of* origin 3.5) (*of* angle 0°) (*with* length 2)) *same*
(*a* Segment (*of* origin 5.5) (*of* angle 180°) (*with* length 2))

It is not allowed to form a description of the same segment by extracting attribution from these two descriptions. In fact

(*a* Segment (*of* origin 3.5) (*of* direction 180) (*with* length 2))

conceivably describes a different segment.

The length of the segment instead is an independent attribute in that it is uniquely determined by the segment (or each of the segments) being described.

One of the goals in the design of Omega has been the possibility to allow for partial descriptions and incremental refinement of description. For example suppose we know that Peter has a child whose name is John:

Peter *is* (a Person (*with* child John))

Later on we learn that Peter has also a daughter:

Peter *is* (a Person (*with* child (*a* Female)))

Then we can deduce:

Peter *is* ((a Person (*with* child John)) *and* (a Person (*with* child (*a* Female))))

From this we would like to conclude that:

Peter *is* (a Person (*with* child John) (*with* child (*a* Female)))

However we do not want the same possibility of merging for *of* attributions. For example it is true that:

6 *is* (*a* Quotient (*of* dividend 12)) *and* (*a* Quotient (*of* divisor 3))

but not

6 *is* (*a* Quotient (*of* dividend 12) (*of* divisor 3))

because:

(*a* Quotient (*of* dividend 12) (*of* divisor 3)) *is* 4

and by transitivity of *is* we would get:

6 *is* 4

Another way of looking at this, is by considering how these examples would be expressed in Predicate Logic:

$(\exists x . 6 = 12/x) \wedge (\exists x . 6 = x/3)$

which is different from saying that:

6 = 12/3

In a sense, attributions replace Skolem functions, in this case the functions **divide** and **multiply**:

6 = 12/divide (12, 6) ∧ 6 = multiply (6, 3)/3

Note that in general several individual descriptions are allowed as attributes for the same attribute name, as in:

(*a* Car (*with* driver John) (*with* driver Jane))

As a consequence it seems convenient to introduce a notation for attributions allowing a unique individual attribute like it is for example the relation **mother** for a person. We will express this fact by using the kind of attribution ***with unique***. So we will say:

(*a* Person (*with unique* mother Sarah))

In addition a notation will be introduced for the possibility to describe any individual attribute having the same attribute name, like for example when we want to describe a person who owns just american cars. For this purpose we will introduce the ***with every*** kind of attributions.

(*a* Driver (*with every* car (*an* American-car)))

It turns out however that the only primitive kind of attribution is ***of***. For this reason it will be enough to give a formal semantics to ***of***. The other kind of attributions (***with***, ***with every*** and ***with unique***) will be defined in terms of ***of*** and properties for each one of them derived as theorems.

The use of universally quantified description variables enables general facts to be asserted about descriptions. Such variables are written as an identifier prefixed by an equal sign symbol. For example we can say that:

(*a* Teacher (*with* subject =x)) *is* (*an* Expert (*with* field =x))

from which we can deduce, by instantiation:

(*a* Teacher (*with* subject music)) *is* (*an* Expert (*with* field music))

Description abstraction is a construct that we introduce in Omega to extend the expressive power of the system. An example of description abstraction is the following:

(*Any* =x *such that* (Carl *is* (*a* Teacher (*with* student =x))))

which denotes the set of individuals for which the predication **(Carl *is* (*a* Teacher (*with* student =x)))** is true, i.e. all the students of Carl.

An extensive discussion on the inference rules for description abstraction will be found in section 5.4.

Omega can be used as a type system. For example it allows to express in a natural way enumerative types and subtypes as shown in the following examples:

(*a* Color) *is* (red *or* green *or* yellow)

(*a* Season) *is* (winter *or* spring *or* summer *or* autumn)

(*a* Positive-Integer) *is* ((*a* Natural-Number) *and* (*not* 0))

(*an* Ordered-Pair (*with* first =x) (*with* second =y))
is (*a* Pair (*with* first (=x *and* (*a* Number))) (*with* second (=y *and* (*a* Number (*with* > =x))))

A description system should be able to express the distinction among the facts that information is missing, that something does not exist or that inconsistent information has been supplied.

The following examples show how Omega deals with these three situations.

A person who has a telephone number will be described as:

(*a* Person (*with* telephone# *Something*))

Someone who does not have a telephone:

not (*a* Person (*with* telephone# *Something*))

A person whose telephone number has been described inconsistently:

(*a* Person (*with* telephone# *Nothing*))

In the TAXIS data model [20] a special constant is provided for each one of these situations (UNKNOWN, NOTHING, INCONSISTENT respectively).

3. Syntax

The language of the theory Omega is presented here using the well established conventional notation of denotational semantics. The syntactic categories of the language are listed below. The rules are presented about how to form elements of such categories from elements of other categories and an infinite class of identifiers. For each category we also show our choice of metavariables ranging on the elements of that category, that we will use in the rest of the exposition.

Category	Syntax	Meta-variables
Ic	identifier	individual constants ($\mathbf{i}$, $\mathbf{i_1}$, $\mathbf{i_2}$ ...)
V	=identifier	variables ($\mathbf{v}$, $\mathbf{v_1}$, $\mathbf{v_2}$ ...)
Cc	identifier	class constants or concepts ($\mathbf{c}$, $\mathbf{c_1}$, $\mathbf{c_2}$ ...)
A	identifier	attribute names ($\mathbf{a_1}$, $\mathbf{a_2}$ ... $\mathbf{b_1}$, $\mathbf{b_2}$)
Ats	(*of* a_1 δ_1) ... (*of* a_n δ_n), $n \geq 0$	attribution sequences (α, α_1, α_2 ...)

Σ	see below	statements (σ, σ_1, σ_2 ...)
Δ	see below	descriptions (δ, δ_1, δ_2 ...)

Descriptions and statements are built from constants and variables according to the following syntax:

Descriptions	Statements
i, *Nothing*, *Something*	*true*, *false*
v	v
$(a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_n\ \delta_n))$, $n \geq 0$	$(\delta_1\ is\ \delta_2)$
(*Any* v *such that* σ)	$\forall v\ .\ \sigma$
$(\delta_1\ or\ \delta_2)$	$(\sigma_1 \vee \sigma_2)$
$(\delta_1\ and\ \delta_2)$	$(\sigma_1 \wedge \sigma_2)$
(*not* δ)	$(\neg\ \sigma)$
σ	$(\sigma_1 \Rightarrow \sigma_2)$

Nothing, ***Something*** are two special constants. ***true*** and ***false*** are individual constants, which are used as truth values.

Note that each statement is also a description, specifically a description of a truth value. Descriptions and statements are mutually recursively defined. Because of this unusual situation, the proof of consistency cannot be done normally by induction on the structure of statements, but is rather performed by induction on the structure of descriptions.

4. Semantics

The method we follow in this work is to define validity semantically. We first characterize a class of models for Omega and define the notion of truth in a model. This provides an intuitive and immediate semantic interpretation for our theory.

We then look for an axiomatization for valid formulas of the theory. The models provided earlier are useful in this stage for providing a guideline and a criterion for suitability of the axiomatization.

By proving the completeness theorem, we show that the axiomatization is adequate. Furthermore the completeness result tells that our theory is consistent if and only if it has a model. Therefore the existence of models presented in this section implies the consistency of Omega.

The structure of the interpretation is

$$\mathcal{A} = \langle I, D, R, \mathfrak{I}, \mathfrak{C} \rangle$$

where I is a non empty set of individuals; D is a family of sets over I, which includes all singletons over I and is closed under the operations of union, intersection and complement; and

$$R \subseteq \mathcal{P}(\bigcup_{n=0}^{\infty} I \times (A \times I)^n)$$

$$\mathfrak{I} : Ic \rightarrow I$$

$$\mathfrak{C} : Cc \rightarrow R$$

$\mathfrak{C}$ is a function, that given a concept returns a set of tuples. Each tuple has an individual as its first component, while the remaining components are name-value pairs representing attributes. The names in such pairs are taken from the set A of attribute names, values are individuals from the domain of interpretation.

This representation has been chosen to express that:

- an individual might have more then one value for the same attribute (i.e. attributes may represent relations rather then functions);
- attributes can have interrelated values (i.e. relations of arity greater then two);
- each individual might have a different set of attributions from other elements in the same class.

We also require that D has the following additional closure properties, needed to model selection by attributes and description abstraction:

$$\forall r \in R, s_1,...,s_n \in D, a_1,...,a_n \in A, \quad (1)$$
$$\{x \in I \mid \exists\ y_1,...,y_n \in I, t \in r\ .\ \forall\ 1 \leq i \leq n\ .\ t = \langle x\ ...\ \langle a_i\ y_i \rangle\ ... \rangle \wedge y_i \in s_i\} \in D$$

$$\{x \in I \mid P(\{x\})\} \in D \quad (2)$$

where P is a formula built using predicate symbol $\subseteq$, the standard logical connectives and quantifiers and variables ranging over elements of D.

4.1. Definition of Value of a Description

The value of a description containing free variables can be determined once a value is assigned to each of those varibles. An environment is a mapping that defines an association between variables and elements of D. The domain of environments is:

$$E : V \rightarrow D$$

We will use the meta-variable ρ for denoting environments.

The value of a description can be defined as a mapping from descriptions and environments into D.

$\mathcal{V}: \Delta \rightarrow E \rightarrow D$

Value of constant descriptions:

a. $\mathcal{V}[\![i]\!]\rho = \{\mathcal{I}[\![i]\!]\}$

b. $\mathcal{V}[\![\textit{Nothing}]\!]\rho = \{\}$

c. $\mathcal{V}[\![\textit{Something}]\!]\rho = I$

$\mathcal{V}[\![v]\!]\rho = \rho(v)$

$\mathcal{V}[\![(a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k))]\!]\rho =$
$\{x \in I \mid \exists\, t \in \mathcal{C}[\![c]\!], y_1, \ldots, y_n \in I\,.\, \forall\, 1 \leq i \leq n\,.\, t = \langle x \ldots \langle a_i\ y_i\rangle \ldots\rangle \wedge y_i \in \mathcal{V}[\![\delta_i]\!]\rho\}$

$\mathcal{V}[\![(Any\ v\ such\ that\ \sigma)]\!]\rho = \{x \in I \mid \models_{\mathcal{A}} [\![\sigma]\!]\rho[\{x\}/v]\}$[1]

$\mathcal{V}[\![(\delta_1\ or\ \delta_2)]\!]\rho = \mathcal{V}[\![\delta_1]\!]\rho \cup \mathcal{V}[\![\delta_2]\!]\rho$

$\mathcal{V}[\![(\delta_1\ and\ \delta_2)]\!]\rho = \mathcal{V}[\![\delta_1]\!]\rho \cap \mathcal{V}[\![\delta_2]\!]\rho$

$\mathcal{V}[\![(not\ \delta)]\!]\rho = I - \mathcal{V}[\![\delta]\!]\rho$

$\mathcal{V}[\![\sigma]\!]\rho = \mathcal{V}[\![true]\!]\rho$ if $\models_{\mathcal{A}} [\![\sigma]\!]\rho$, $\mathcal{V}[\![false]\!]\rho$ if $\not\models_{\mathcal{A}} [\![\sigma]\!]\rho$

The definition of an instance description says that all tuples from the class associated to **c** are considered which have, for each attribution, a component which has the name of that attribution as a tag and a value which satisfies the corresponding description of the attribute. The set of the first elements of such tuples is the value of the instance description.

As an example we show how to determine the value of the instance description **(*a* Product (*of* factor$_1$ 2))**. Let us suppose that:

$\mathcal{C}$ [[Product]] =
{<0 <factor$_1$ 0> <factor$_2$ 0>>, <0 <factor$_1$ 0> <factor$_2$ 1>>, <0 <factor$_1$ 0> <factor$_2$ 2>>, ... ,
<1 <factor$_1$ 1> <factor$_2$ 1>>,
<2 <factor$_1$ 2> <factor$_2$ 1>>, <2 <factor$_1$ 1> <factor$_2$ 2>>,
<3 <factor$_1$ 3> <factor$_2$ 1>>, <3 <factor$_1$ 1> <factor$_2$ 3>>, ...}

$\mathcal{V}[\![a\ \text{Product}\ (of\ \text{factor}_1\ 2)]\!] = \{x \in D \mid \exists\, y_1, \ldots, y_n \in D, t \in \mathcal{C}[\![\text{Product}]\!]\,.$
$t = \langle x \ldots \langle \text{factor}_1\ y_1\rangle \ldots\rangle \wedge y_1 \in \{2\}\} = \{0, 2, 4, 6 \ldots\}$

D → I

Note that an attribution named a_1 in an instance description δ implies that for any individual **i** described by δ there exist some individual which is the value for the attribute a_1 of i. The exception is when the attribute is

[1] $\rho[x/v]$ is the same environment as ρ but for variable v to which it associates x

Nothing. In this case the whole instance description reduces to *Nothing*. When a partial description is formed omitting some of the attributes, its interpretation is that omitted attributes range over all possible values.

The interpretation of the description abstraction ***Any*** is the set of individuals which, substituted for v in the statement σ, make the statement true. The notation

$$\models_{\mathcal{A}} [\![\sigma]\!]\rho$$

means that the statement σ in the *environment* ρ is true relative to the structure $\mathcal{A}$, and it is completely defined in the next section.

4.2. Definition of Truth Value

1. $\models_{\mathcal{A}} [\![\mathit{true}]\!]\rho$, $\models_{\mathcal{A}} [\![\neg \mathit{false}]\!]\rho$

2. Variables

   a. $\models_{\mathcal{A}} [\![v]\!]\rho$ iff $\rho(v) = \Upsilon[\![\mathit{true}]\!]\rho$

   b. $\models_{\mathcal{A}} [\![\neg v]\!]\rho$ iff $\rho(v) = \Upsilon[\![\mathit{false}]\!]\rho$

3. $\models_{\mathcal{A}} [\![(\delta_1 \mathit{\ is\ } \delta_2)]\!]\rho$ iff $\Upsilon[\![\delta_1]\!]\rho \subseteq \Upsilon[\![\delta_2]\!]\rho$

4. $\models_{\mathcal{A}} [\![(\delta_1 \mathit{\ same\ } \delta_2)]\!]\rho$ iff $\Upsilon[\![\delta_1]\!]\rho = \Upsilon[\![\delta_2]\!]\rho$

5. $\models_{\mathcal{A}} [\![(\forall v\ .\ \sigma)]\!]\rho$ iff for all $x \in D$, $\models_{\mathcal{A}} [\![\sigma]\!]\rho[x/v]$

6. $\models_{\mathcal{A}} [\![(\sigma_1 \vee \sigma_2)]\!]\rho$ iff $\models_{\mathcal{A}} [\![\sigma_1]\!]\rho$ or $\models_{\mathcal{A}} [\![\sigma_2]\!]\rho$

7. $\models_{\mathcal{A}} [\![(\sigma_1 \wedge \sigma_2)]\!]\rho$ iff $\models_{\mathcal{A}} [\![\sigma_1]\!]\rho$ and $\models_{\mathcal{A}} [\![\sigma_2]\!]\rho$

8. $\models_{\mathcal{A}} [\![\neg \sigma]\!]\rho$ iff not $\models_{\mathcal{A}} [\![\sigma]\!]\rho$

9. $\models_{\mathcal{A}} [\![(\sigma_1 \Rightarrow \sigma_2)]\!]\rho$ iff $\models_{\mathcal{A}} [\![\sigma_1]\!]\rho$ implies $\models_{\mathcal{A}} [\![\sigma_2]\!]\rho$

The truth of a statement relative to a structure $\mathcal{A}$ is defined as:

Definition 1: (Truth) $\quad \models_{\mathcal{A}} \sigma$ iff $\forall \rho \models_{\mathcal{A}} [\![\sigma]\!]\rho$

Given the definition of truth, validity is defined as follows:

Definition 2: (Validity) $\quad \models \sigma$ iff $\forall \mathcal{A} \models_{\mathcal{A}} \sigma$

5. Axiomatization

We turn now to the question of finding an axiomatization of the universally valid formulas.

The method that we follow in deriving the axiomatization, is based on noting rules that preserve validity of

formulae and turning them into axioms or inference rules.

We present the axioms of the system in the form of natural deduction inference rules [13]. As a matter of notation we write

$$\frac{\sigma_1, \ldots, \sigma_n}{\sigma}$$

meaning that from the premises $\sigma_1, \ldots, \sigma_n$ we can derive the consequence σ and that from the premise σ we can derive any of the consequences $\sigma_1, \ldots, \sigma_n$. We omit the line when the set of premises is empty.

The inference rules will be written as

$$\frac{[\sigma_0]}{\sigma_1, \ldots, \sigma_n}$$

$$\sigma$$

with the meaning that σ can be derived from $\sigma_1, \ldots, \sigma_n$, and if σ_0 was used as a premise in deriving any one of $\sigma_1, \ldots, \sigma_n$, it will not appear as a premise for the conclusion σ.

5.1. Axioms for Descriptions

The notion of "sameness" between descriptions is defined as follows:

Definition 3: (Sameness)

$(\delta_1 \textit{ same } \delta_2)$ iff $((\delta_1 \textit{ is } \delta_2) \wedge (\delta_2 \textit{ is } \delta_1))$

In the following we will also use the following abbreviation:

Individual $[\delta]$ to stand for $\neg(\delta \textit{ is Nothing}) \wedge \forall \nu . (\nu \textit{ is } \delta) \Rightarrow (\delta \textit{ is } \nu) \vee (\nu \textit{ is Nothing})$

Individuals are those descriptions who are not ***Nothing*** and are at the lowest level in the lattice of descriptions: only ***Nothing*** is below them. Intuitively, the notion of individual corresponds to description that cannot be further specified. In our set theoretical interpretation, individuals correspond to singletons, i.e. sets consisting of a single element.

Axioms for Descriptions

D1: $\delta_1 \textit{ is } \delta_2 \Leftrightarrow (\forall\, \mathrm{v}\,.\,(\mathrm{Individual}[\mathrm{v}] \wedge \mathrm{v} \textit{ is } \delta_1) \Rightarrow \mathrm{v} \textit{ is } \delta_2)$

D2: δ *is Something*

D3: *Nothing is* δ

D4: *Something is* $(\delta$ *or* $(\textit{not } \delta))$

D5: $(\delta$ *and* $(\textit{not } \delta))$ *is Nothing*

D6: $$\frac{\delta_1 \textit{ is } \delta_2,\ \delta_1 \textit{ is } \delta_3}{\delta_1 \textit{ is } (\delta_2 \textit{ and } \delta_3)}$$

D7: $$\frac{\delta_1 \textit{ is } \delta_3,\ \delta_2 \textit{ is } \delta_3}{(\delta_1 \textit{ or } \delta_2) \textit{ is } \delta_3}$$

D8: $$\frac{\delta_1 \textit{ is } \delta_2}{\textit{not } \delta_2 \textit{ is not } \delta_1}$$

D9: $\textit{not } (\textit{not } \delta) \textit{ is } \delta$

D10: $\delta \textit{ is not } (\textit{not } \delta)$

Axiom **D1** is a version of the axiom of extensionality for sets, phrased in a way to take into account that only sets consisting of individuals are being considered.

5.2. Axioms for Statements

Axioms for Statements

S1: true

S2: $\neg\, \mathit{false}$

S3: $\sigma \Rightarrow \mathit{true}$

S4: $\mathit{false} \Rightarrow \sigma$

S5: $\mathit{true} \Rightarrow \sigma \vee \neg\, \sigma$

S6: $\sigma \wedge \neg\, \sigma \Rightarrow \mathit{false}$

S7:
$$\frac{\sigma_1 \Rightarrow \sigma_2,\ \sigma_1 \Rightarrow \sigma_3}{\sigma_1 \Rightarrow \sigma_2 \wedge \sigma_3}$$

S8:
$$\frac{\sigma_1 \Rightarrow \sigma_3,\ \sigma_2 \Rightarrow \sigma_3}{\sigma_1 \vee \sigma_2 \Rightarrow \sigma_3}$$

S9:
$$\frac{\sigma_1 \Rightarrow \sigma_2}{\neg\sigma_2 \Rightarrow \neg\sigma_1}$$

S10: $\neg\, (\neg\sigma) \Rightarrow \sigma$

S11: $\sigma \Rightarrow \neg\, (\neg\sigma)$

S12: $\sigma \Leftrightarrow \sigma$ *same true*

S13: $\neg\sigma \Leftrightarrow \sigma$ *same false*

Note that this axiom set is not minimal, since for instance S1-S2, S3-S4 and S5-S6 are pairwise derivable from one another. A similar remark applies to the axioms for descriptions presented before.

We have chosen this set of axioms so that there is an almost complete symmetry between the axioms for statements and the axioms for descriptions. In this way any theorem about descriptions has a corresponding dual theorem about statements. A single proof procedure will work for both descriptions and statements.

Axioms S12 and S13 are necessary in our assumption that statements are descriptions. The first one for example allows a statement to be asserted whenever it describes the truth value ***true***.

There is a strict correspondence between statements and descriptions, which is expressed in the following

lemma:

Lemma 4:

1. ***Nothing same*** **(*Any*** v ***such that false*)**

2. ***Something same*** **(*Any*** v ***such that true*)**

3. **(δ_1 *or* δ_2) *same* (*Any* v *such that* (v *is* δ_1) $\vee$ (v *is* δ_2))**

4. **(δ_1 *and* δ_2) *same* (*Any* v *such that* (v *is* δ_1) $\wedge$ (v *is* δ_2))**

5. **(*not* δ) *same* (*Any* v *such that* $\neg$ (v *is* δ))**

Actually these statements could have as well been given as an alternative axiomatization for description operators. We prefer however the set of axioms presented above since they do not involve explicitly description abstraction. As we will see, the inference rules for description abstraction had to be restricted to apply only on individuals. With the axioms we have chosen, the use of abstraction rules and extensionality (axiom D1) can often be avoided.

5.3. Axioms for Attributions

It is easy to verify that the following axioms of *Commutativity*, *Omission*, *Monotonicity* and *Strictness* hold under the interpretation given for ***of*** attributions.

The axiom of *Commutativity* expresses the fact that the order of attributions is irrelevant, and can be stated as follows:

(a c α_1 (*of* a_1 δ_1) α_2) *is* (a c (*of* a_1 δ_1) α_1 α_2) (*Commutativity*)

For example:

(*a* Queue (*of* front (*an* Integer)) (*of* rear (*a* List-of-Integers))) *is*
(*a* Queue (*of* rear (*a* List-of-Integers)) (*of* front (*an* Integer)))

The axiom of omission states the fact that if we omit an attribution we get a more general description:

(a c (*of* a_1 δ) α) *is* (a c α) (*Omission*)

For example

(*a* Student (*of* curriculum Engineering) (*of* year 2)) *is* (*a* Student (*of* curriculum Engineering))

The axiom of *Monotonicity* is stated as follows:

δ_1 *is* δ_2 $\Rightarrow$ (a c (*of* a_1 δ_1) α) *is* (a c (*of* a_1 δ_2) α) (*Monotonicity*)

For example:

John *is* (*an* Artist) ⇒ ((*a* Person (*of* child John)) *is* (*a* Person (*of* child (*an* Artist))))

(*a* Factorial (*of* arg 0)) *is* 1 ⇒ (*a* Product (*of* arg_1 (*a* Factorial (*of* arg 0)))) *is* (*a* Product (*of* arg_1 1))

The axiom of *Strictness* corresponds to the intuitive notion that when an inconsistent description is given for an attribute then the whole description is inconsistent, i.e. it does not describe anything. The formal statement of the axiom is as follows:

(*a* c (*of* a_1 *Nothing*)) *is Nothing* (*Strictness*)

For example:

(*a* Square (*of* arg (6 *and* (*not* 6)))) *is* (*a* Square (*of* arg *Nothing*))

(*a* Square (*of* arg *Nothing*)) *is Nothing*

5.4. Inference Rules

It is easy to verify that these two *transitivity* rules preserve the truth according to our interpretation:

$$\frac{\sigma_1 \Rightarrow \sigma_2,\ \sigma_2 \Rightarrow \sigma_3}{\sigma_1 \Rightarrow \sigma_3} \qquad \frac{\delta_1\ \textit{is}\ \delta_2,\ \ \delta_2\ \textit{is}\ \delta_3}{\delta_1\ \textit{is}\ \delta_3}$$

The ordinary rule of Modus Ponens can be derived from transitivity by letting σ_1 be ***true***.

We need a rule for ⇒ introduction:

$$\frac{\begin{array}{c}[\sigma_1]\\ \sigma_2\end{array}}{\sigma_1 \Rightarrow \sigma_2}$$

Next note that the rule of generalization is sound:

$$\frac{\sigma[v']}{\forall\, v\,.\,\sigma[v]}$$

where v is a new variable, or at least one that does not appear in any assumption on which σ depends.

Also sound is the corresponding instantiation rule:

$$\frac{\forall\, v\,.\,\sigma[v]}{\sigma[\delta]}$$

The most natural formulation of the abstraction rule for descriptions would be:

$$\frac{\sigma[\delta]}{\delta \textit{ is } (\textit{Any } \nu \textit{ such that } \sigma[\nu])} \quad \textbf{A1}$$

and the corresponding concretion rule:

$$\frac{\delta \textit{ is } (\textit{Any } \nu \textit{ such that } \sigma[\nu])}{\sigma[\delta]} \quad \textbf{C1}$$

Unrestricted abstraction and concretion rules lead to a number of problems with the interpretation of descriptions given above. Let us consider for example

John *is* (*a* Human)

With **(*a* Human)** for δ, by A1, we get:

(*a* Human) *is* (*Any* ν *such that* (John *is* ν))

but if also

Paul *is* (*a* Human)

then by transitivity:

Paul is (*Any* ν *such that* (John *is* ν))

and by C1:

(John *is* Paul)

which might not be true.

A restricted form of the abstraction principle that avoids such problems is the following:

$$\frac{\sigma[\delta] \wedge \textbf{Individual}[\delta]}{\delta \textit{ is } (\textit{Any } \nu \textit{ such that } \sigma[\nu])} \quad \textbf{A2}$$

where ν is a new variable. Similar problems arise with an unrestricted concretion rule. Let $\sigma[\nu]$ = **Individual[ν]** and suppose:

Individual[John]
Individual[Paul]

then by A2

John *is* (*Any* ν *such that* Individual[ν])
Paul *is* (*Any* ν *such that* Individual[ν])

From the axiom D6, which introduces the ***or*** description, we get:

(John *or* Paul) *is* (*Any* ν *such that* Individual[ν])

and from this and C1:

Individual [John *or* Paul]

We solve this problem by restricting C1 analogously to what we did for A1:

$$\frac{\delta \textit{ is } (\textit{Any } \nu \textit{ such that } \sigma[\nu]) \wedge \text{Individual}[\delta]}{\sigma[\delta]} \qquad \textbf{C2}$$

With this restriction to individuals, it is easy to verify that the abstraction and concretion rules are sound.

Note that with this form of the concretion principle we avoid Russell's paradox. Suppose that **z** is the following description:

$$z \equiv (\textit{Any } \nu \textit{ such that } \neg (\nu \textit{ is } \nu))$$

If we allow C1, since $z \textit{ is } z$ is true by reflexivity of ***is***, we would derive the paradoxical consequence:

$$\neg(z \textit{ is } z)$$

We can show however that:

$$z \textit{ same Nothing}$$

because

$$(\textit{Any } \nu \textit{ such that } \neg (\nu \textit{ is } \nu)) \textit{ same } (\textit{Any } \nu \textit{ such that false}) \textit{ same Nothing}$$

Therefore rule C2 cannot be applied because ***Nothing*** is not an individual, as it follows from the definition of individual.

The complete set of inference rules is summarized in the following table.

Inference Rules

Statements	Descriptions
$\dfrac{[\sigma_1]\ \sigma_2}{\sigma_1 \Rightarrow \sigma_2}$	
$\dfrac{\sigma_1 \Rightarrow \sigma_2,\ \sigma_2 \Rightarrow \sigma_3}{\sigma_1 \Rightarrow \sigma_3}$	$\dfrac{\delta_1\ \textit{is}\ \delta_2,\ \delta_2\ \textit{is}\ \delta_3}{\delta_1\ \textit{is}\ \delta_3}$
$\dfrac{\sigma[v']}{\forall\ v\ .\ \sigma[v]}$	$\dfrac{\sigma[\delta] \wedge \text{Individual}[\delta]}{\delta\ \textit{is}\ (\textit{Any}\ v\ \textit{such that}\ \sigma[v])}$
$\dfrac{\forall\ v\ .\ \sigma[v]}{\sigma[\delta]}$	$\dfrac{\delta\ \textit{is}\ (\textit{Any}\ v\ \textit{such that}\ \sigma[v]) \wedge \text{Individual}[\delta]}{\sigma[\delta]}$

5.5. Theorems and Derivation

A derivability relation $\vdash$ is defined as usual, so that $\Gamma \vdash \sigma$ means that the statement σ can be derived from the statements in Γ and from the axioms by applying the rules of inference.

We list here some theorems of Omega that are needed in later proofs and give an example of a derivation.

Lemma 5:

1. δ *is* δ
2. (δ_1 *and* δ_2) *same not* (*not* δ_1 *or not* δ_2)
3. (δ_1 *or* δ_2) *same not* (*not* δ_1 *and not* δ_2)
4. (δ_1 *is* ((δ_2 *or* δ_3) *and not* δ_2)) $\Rightarrow$ (δ_1 *is* δ_3)

5. $\neg(\delta_1$ *is Nothing*$) \Rightarrow (\delta_1$ *is not* $\delta_2 \Leftrightarrow \neg(\delta_1$ *is* $\delta_2))$

6. Individual $[\delta] \Rightarrow (\delta$ *is* $(\delta_1$ *or* $\delta_2) \Leftrightarrow (\delta$ *is* $\delta_1) \vee (\delta$ *is* $\delta_2))$

7. Individual $[\delta] \Rightarrow ((\delta$ *is* $($*not* $\delta_1)) \Leftrightarrow \neg(\delta$ *is* $\delta_1))$

8. $\sigma_1 \Rightarrow \sigma_1 \vee \sigma_2$

9. Individual $[\delta_1] \wedge \neg ((\delta_1$ *and* $\delta_2)$ *is Nothing*$) \Rightarrow (\delta_1$ *is* $\delta_2)$

The second and third statements correspond to the deMorgan's laws. As an example of deduction in Omega we prove the first of the De Morgan's laws, namely statement 2.

Proof: From reflexivity we have:

1. $(\delta_1$ *and* $\delta_2)$ *is* $(\delta_1$ *and* $\delta_2)$

By applying axiom D7 to this, we get:

2. $(\delta_1$ *and* $\delta_2)$ *is* δ_1

Then, by complementation (axiom D9):

3. $($*not* $\delta_1)$ *is not* $(\delta_1$ *and* $\delta_2)$

In a completely similar way we get also:

4. $($*not* $\delta_2)$ *is not* $(\delta_1$ *and* $\delta_2)$

Then, from 3 and 4 we can introduce the *or*, using axiom D8:

5. $($*not* δ_1 *or not* $\delta_2)$ *is not* $(\delta_1$ *and* $\delta_2)$

and finally, again by complementation:

6. $(\delta_1$ *and* $\delta_2)$ *is not* $($*not* δ_1 *or not* $\delta_2)$

The inclusion in the other direction is proved similarly.

6. Soundness and Completeness

The first result to be proved about the axiomatization is its soundness:

Theorem 6: (Soundness) For every closed statement A, $\Gamma \vdash A \Rightarrow \Gamma \models A$

We are using the notation $\Gamma \models A$ as an abbreviation for:

$$\forall M . (\forall \sigma . \sigma \in \Gamma \Rightarrow \models_M \sigma) \Rightarrow \models_M A$$

The soundness theorem states that whenever a statement can be derived from the set of premises Γ then it is true in every model that satisfies the statements in Γ.

Proof: Omitted since the argument is straightforward.

We next turn to prove the completeness theorem for the theory of Omega.

The completeness theorem gives us a measure of the adequacy of our axiomatization. In fact it asserts that the set of valid formulas coincides with the set of theorems of our theory. In this way it provides a bridge between what is established as semantically valid, and what is established by syntactic symbolic manipulations.

More important, this result is the fundamental step in showing the consistency of our formal theory.

The completeness theorem can be formulated as follows:

Theorem 7: (Completeness) For every closed statement A, $\Gamma \models A \Leftrightarrow \Gamma \vdash A$

which means that any formula true in every model that satisfies the set of premises Γ, can be derived from Γ by the rules of Omega and vice versa.

We present the general outline of the proof, which follows the lines of a Goedel-Henkin argument [24].

Sketch of the proof. The implication $\Gamma \vdash A \Rightarrow \Gamma \models A$ corresponds to the soundness theorem. The other direction of the implication is equivalent to saying:

$$\Gamma \nvdash A \Rightarrow \exists M \; \Gamma \not\models_M A$$

If A cannot be derived from Γ then there exists a model of Γ in which A does not hold.

In order to find such a model, for a given closed formula A, we build a complete Henkin extension of theory Γ, called Γ_A, having the property that:

$$\Gamma \nvdash A \Rightarrow \Gamma_A \nvdash A \tag{3}$$

Being a complete, Γ_A will have the property:

$$\forall \text{ closed } \sigma \text{ either } \Gamma_A \vdash \sigma \text{ or } \Gamma_A \vdash \neg \sigma \tag{4}$$

That Γ_A is a Henkin theory means that:

For every statement $\sigma[v]$, there is a constant c, such that: $\Gamma_A \vdash (\sigma[c] \Rightarrow \forall v \,.\, \sigma[v])$ (5)

Moreover, since there is an implicit existential quantifier in the attribution notation, our definition of a Henkin theory will also require that:

For every statement of the form $(\delta \; is \; (a \; c \; (of \; a_1 \; \delta_1) \ldots (of \; a_n \; \delta_n)))$, (6)
there exist n individual constants $\gamma_1, \ldots, \gamma_n$ such that:

$$\Gamma_A \vdash \delta \; is \; (a \; c \; (of \; a_1 \; \delta_1) \ldots (of \; a_n \; \delta_n)) \wedge \text{Individual}[\delta] \Rightarrow$$
$$\delta \; is \; (a \; c \; (of \; a_1 \; \gamma_1) \ldots (of \; a_n \; \gamma_n)) \wedge (\gamma_1 \; is \; \delta_1) \wedge \ldots (\gamma_n \; is \; \delta_n)$$

Next we will prove the following lemma in order to obtain a model for Γ_A:

Lemma 8: (Main lemma)

$$\exists M . \forall \sigma (\Gamma_A \vdash \sigma \Leftrightarrow \Gamma_A \models_M \sigma)$$

Given such a model $\mathcal{M}$ for Γ_A, we assume that $\Gamma \nvdash A$ and note that from property 3 it follows that $\Gamma_A \nvdash A$. Then:

$$\Gamma_A \not\models_{\mathcal{M}} A$$

Since a model of Γ_A can be restricted to a model of Γ, we have proved that:

$$\Gamma \nvdash A \Rightarrow \exists M . \Gamma \not\models_M A$$

and the completeness theorem.

6.1. Construction of the Complete Henkin Extension

We will assume from now on that $\Gamma \nvdash A$. As a consequence we are assuming that Γ is consistent.

We build a Henkin theory Γ' as the limit of an inductive series of theories defined as follows:

$\Gamma_0 = \Gamma$. The *special constants of level 0* are defined as the empty set.

Γ_{n+1} is obtained by adding to Γ_n:

- the statement $\sigma[\gamma] \Rightarrow \forall v . \sigma[v]$, where γ is a new constant, for each statement of the form $\forall v . \sigma[v]$, with occurrences of special constants of level n.

- the statement

$$\delta \textit{ is } (a \text{ c } (\textit{of } a_1 \ \delta_1) \ldots (\textit{of } a_k \ \delta_k)) \wedge \text{Individual}[\delta] \Rightarrow$$
$$\delta \textit{ is } (a \text{ c } (\textit{of } a_1 \ \gamma_1) \ldots (\textit{of } a_k \ \gamma_k)) \wedge (\gamma_1 \textit{ is } \delta_1) \wedge \ldots (\gamma_k \textit{ is } \delta_k)$$

  where γ_1, ... γ_k are new constants, for each statement of the form $(\delta \textit{ is } (a \text{ c } (\textit{of } a_1 \ \gamma_1) \ldots (\textit{of } a_k \ \gamma_k)))$, with occurrences of special constants of level n. All the constants added during this step constitute the *special constants of level n+1*.

Definition 9: $\Gamma' = \cup \Gamma_n$.

Lemma 10: Γ' is a conservative extension of Γ, i.e.

If σ contains no constants but those in Γ and $\Gamma' \vdash \sigma$, then $\Gamma \vdash \sigma$ (7)

Proof: Suppose $\Gamma' \vdash \sigma$, and the proof uses just one premise from Γ' that is not in Γ, namely σ'. Then $\Gamma \vdash \sigma' \Rightarrow \sigma$. The statement σ' can be in one of these two forms:

Case 1. $\sigma' \equiv \sigma_1[\gamma] \Rightarrow \forall v . \sigma_1[v]$

$\Gamma \vdash (\sigma_1[\gamma] \Rightarrow \forall v . \sigma_1[v]) \Rightarrow \sigma$
$\Gamma \vdash (\sigma_1[z] \Rightarrow \forall v . \sigma_1[v]) \Rightarrow \sigma$
since γ is a constant not present in Γ and z is a new variable.
$\Gamma \vdash (\forall z . \sigma_1[z] \Rightarrow \forall v . \sigma_1[v]) \Rightarrow \sigma$
But since $\Gamma \vdash \forall z . \sigma_1[z] \Rightarrow \forall v . \sigma_1[v]$, by *Modus Ponens* $\Gamma \vdash \sigma$

Case 2. $\sigma' \equiv \delta \textit{ is } (a \text{ c } (\textit{of } a_1 \ \delta_1) \ldots (\textit{of } a_k \ \delta_k)) \wedge \text{Individual}[\delta] \Rightarrow \delta \textit{ is } (a \text{ c } (\textit{of } a_1 \ \gamma_1) \ldots (\textit{of}$

$a_k\ \gamma_k)) \wedge (\gamma_1\ \textit{is}\ \delta_1) \wedge \ldots \wedge (\gamma_k\ \textit{is}\ \delta_k)$

We note that
$\sigma' \Leftrightarrow (\sigma''[\gamma_1, \ldots, \gamma_k] \Rightarrow \forall\ v_1, \ldots, v_k\ .\ \sigma''[v_1, \ldots, v_k])$
where
$\sigma''[v_1, \ldots, v_k] = \neg(\delta\ \textit{is}\ (a\ c\ (\textit{of}\ a_1\ v_1) \ldots (\textit{of}\ a_k\ v_k))) \vee \neg\ \text{Individual}[\delta] \vee$
$\neg\ (v_1\ \textit{is}\ \delta_1) \vee \ldots \vee \neg\ (v_k\ \textit{is}\ \delta_k)$
Therefore this case reduces to case 1.

The general case when σ is proved in Γ' by using n premises not in Γ, can be proved by applying inductively the argument above.

To obtain the complete extension of Γ we are looking for, we exploit a known result:

Theorem 11: (Lindenbaum) If Γ is consistent, then Γ has a complete simple extension.

Since $\Gamma' \nvdash A$, we can now define Γ_A as follows:

Definition 12: Γ_A is any complete simple extension of $\Gamma' \cup \{\neg A\}$.

Lemma 13: Γ_A has properties 3, 4, 5, 6.

Proof: It follows from the definition of Γ_A, and lemma 10.

6.2. Proof of Main Lemma

We will prove the main lemma by building a model $\mathcal{M}_b$ such that:

$\forall\ \sigma\ \Gamma_A \vdash \sigma \Leftrightarrow \Gamma_A \models_{\mathcal{M}_b} \sigma$

$\mathcal{M}_b$ will be a term model built out of syntactic material. More precisely equivalence classes of individual descriptions will be the elements of the domain of interpretation. Let us define the equivalence relation ~ as:

$\delta \sim \delta' \equiv \Gamma_A \vdash (\delta\ \textit{same}\ \delta')$

We will denote as $|\delta|$ the equivalence class of δ according to ~.

We will call $\mathbf{Ind} = \{\delta \in \Delta \mid \Gamma_A \vdash \text{Individual}[\delta]\}$, and $I = \mathbf{Ind}/\sim$, the quotient of the set **Ind** with respect to the equivalence relation ~. Finally, let $D = \{\{|\delta| \in I \mid \Gamma_A \vdash (\delta'\ \textit{is}\ \delta)\} \mid \delta' \in \Delta\}$.

The model $\mathcal{M}_b$ is defined as: $\mathcal{M}_b = \langle I, D, R, \mathcal{I}, \mathcal{C}\rangle$ where

$\mathcal{I}\ [\![i]\!] = |i|$

$\mathcal{C}\ [\![c]\!] = \{\langle|\delta|\ \langle a_1\ |\delta_1|\rangle \ldots \langle a_n\ |\delta_n|\rangle\rangle \mid \delta, \delta_1, \ldots \delta_n \in \mathbf{Ind}, a_1, \ldots a_n \in A,$
$\Gamma_A \vdash \delta\ \textit{is}\ (a\ c\ (\textit{of}\ a_1\ \delta_1) \ldots (\textit{of}\ a_n\ \delta_n))\}$

$R = \{\mathcal{C}\ [\![c]\!] \mid c \in Cc\}$

It is not difficult to show that this model has the required closure properties:

Lemma 14: D contains all singletons of I, is closed with respect to union, intersection, complement and has properties 1 and 2 with respect to R.

With the following lemma we reduce our problem to a property of our the model just defined.

Lemma 15: If $\mathcal{V}[\![\sigma]\!] = \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is } \sigma)\}$ then $\Gamma_A \vdash \sigma \Leftrightarrow \Gamma_A \models_{\mathcal{M}_b} \sigma$

Proof: Assume first that $\Gamma_A \vdash \sigma$. By axiom S12, it is also $\Gamma_A \vdash (\sigma \textit{ is } \textit{true})$. Using this fact and transitivity in the premise of the lemma, we have

$$\mathcal{V}[\![\sigma]\!] \subseteq \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is } \textit{true})\} = \{|\textit{true}|\}$$

But by the definition of $\mathcal{V}$:

$$\mathcal{V}[\![\sigma]\!] = \{|\textit{true}|\} \text{ if and only if } \Gamma_A \models_{\mathcal{M}_b} \sigma$$

On the other hand, if we assume that $\Gamma_A \models_{\mathcal{M}_b} \sigma$ then

$$\mathcal{V}[\![\sigma]\!] = \{|\textit{true}|\} = \{|\delta'| \mid \delta' \in \mathbf{Ind} \text{ and } \Gamma_A \vdash (\delta' \textit{ is } \sigma)\}$$

This means that $\Gamma_A \vdash (\textit{true is } \sigma)$. Since Γ_A is complete, then either $\Gamma_A \vdash \sigma$ or $\Gamma_A \vdash \neg\, \sigma$. But the latter case is impossible because, by applying axiom S13, we would get $\Gamma_A \vdash (\sigma \textit{ is false})$, and by transitivity $\Gamma_A \vdash (\textit{true is false})$, and since both *true* and *false* are individuals (axiom D2) $\Gamma_A \vdash (\textit{true same false})$, and, by axiom S12, $\Gamma_A \vdash \textit{false}$, which contradicts the consistency assumption for Γ_A.

The model we have defined has the following significant property, which establishes the connection between the semantics (value of descriptions) and the syntax (derivability of predications):

Lemma 16: For every closed description δ:

$$\mathcal{V}[\![\delta]\!] = \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is } \delta)\}$$

Proof: The proof is done by induction on the structure of descriptions.

1. $\delta \equiv \mathbf{i}$.

   $$\mathcal{V}[\![\mathbf{i}]\!] = \{|\mathbf{i}|\} = \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is } \mathbf{i})\}$$

   since $\mathbf{i} \in \mathbf{Ind}$ by axiom D2. Note that this case takes care also of the individual constants ***true*** and ***false***.

2. $\delta \equiv \textit{Nothing}$

   $$\mathcal{V}[\![\textit{Nothing}]\!] = \{\} = \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is Nothing})\}$$

   since ***Nothing*** is not an individual.

3. $\delta \equiv \textit{Something}$

   $$\mathcal{V}[\![\textit{Something}]\!] = \mathbf{Ind}/\sim = \{|\delta'| \mid \delta' \in \mathbf{Ind}, \Gamma_A \vdash (\delta' \textit{ is Something})\}$$

   since δ' ***is Something*** can be proved of any description δ'.

4. $\delta \equiv (a\ \mathbf{c}\ (of\ \mathbf{a}_1\ \delta_1) \ldots (of\ \mathbf{a}_k\ \delta_k))$.

$\mathcal{V}[\![a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k)]\!] =$ (by definition of $\mathcal{V}$)

$\{|\delta| \mid \delta \in \mathbf{Ind}, \exists\, t \in \mathcal{C}(c), \delta'_1, \ldots \delta'_n \in \mathbf{Ind}\ .$
$\forall\, i\ .\ 1 \leq i \leq n\ .\ t = \langle|\delta| \langle \ldots \langle a_i\ |\delta'_i| \rangle \ldots \rangle \wedge |\delta'_i| \in \mathcal{V}[\![\delta_i]\!]\} =$
(by definition of $\mathcal{C}$ and induction hypothesis)

$\{|\delta| \mid \delta \in \mathbf{Ind}, \exists\, \delta' \in \Delta, \delta'_1, \ldots \delta'_n \in \mathbf{Ind}\ .$
$\Gamma_A \vdash \delta\ is\ \delta',$
$\forall\, i\ .\ 1 \leq i \leq n\ \delta' = (a\ c \ldots (of\ a_i\ \delta'_i) \ldots) \wedge \Gamma_A \vdash \delta'_i\ is\ \delta_i\} \subseteq$
(by *Monotonicity* and *Omission*)

$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash \delta\ is\ (a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k))\}$

We can show the inclusion in the other direction by considering two cases:

a) $\Gamma_A \vdash (\delta_1\ is\ Nothing) \vee \ldots \vee (\delta_k\ is\ Nothing)$

In this case, by *Monotonicity* and *Strictness*:
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash \delta\ is\ (a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k))\} \subseteq$
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash \delta\ is\ Nothing\} = \{\}$
(since no individual is ***Nothing***)

b) $\Gamma_A \vdash \neg(\delta_1\ is\ Nothing) \wedge \ldots \wedge \neg(\delta_k\ is\ Nothing)$

$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash \delta\ is\ (a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k))\} \subseteq$
(by property 6 of Γ_A)

$\{|\delta| \mid \delta \in \mathbf{Ind}, \exists\, \delta'_1, \ldots \delta'_k \in \mathbf{Ind}\ .$
$\Gamma_A \vdash \delta\ is\ (a\ c\ (of\ a_1\ \delta'_1) \ldots (of\ a_k\ \delta'_k)), (\delta'_1\ is\ \delta_1) \wedge \ldots \wedge (\delta'_k\ is\ \delta_k)\} \subseteq$

$\mathcal{V}[\![a\ c\ (of\ a_1\ \delta_1) \ldots (of\ a_k\ \delta_k)]\!]$

5. $\delta \equiv (\delta_1\ or\ \delta_2)$.

$\mathcal{V}[\![\delta_1\ or\ \delta_2]\!] =$ (by definition of $\mathcal{V}$)
$\mathcal{V}[\![\delta_1]\!] \cup \mathcal{V}[\![\delta_2]\!] =$ (by induction hypothesis)
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash (\delta\ is\ \delta_1)\} \cup \{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash (\delta\ is\ \delta_2)\} =$
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash (\delta\ is\ \delta_1)$ or $\Gamma_A \vdash (\delta\ is\ \delta_2)\} =$
(by Lemma 5.8 and the completeness of Γ_A)
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash (\delta\ is\ \delta_1) \vee (\delta\ is\ \delta_2)\} =$ (by Lemma 5.6)
$\{|\delta| \mid \delta \in \mathbf{Ind}, \Gamma_A \vdash (\delta\ is\ (\delta_1\ or\ \delta_2))\}$

6. $\sigma \equiv (\delta_1\ is\ \delta_2)$.

Assume first that: $\mathcal{V}[\![\delta_1]\!] \subseteq \mathcal{V}[\![\delta_2]\!]$

$\{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is } \delta_1)\} \subseteq \{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is } \delta_2)\}$

(by induction hypothesis)

$\delta' \in \text{Ind}, \ \Gamma_A \vdash \delta' \textit{ is } \delta_1 \Rightarrow \Gamma_A \vdash \delta' \textit{ is } \delta_2$

$\Gamma_A \vdash \delta_1 \textit{ is } \delta_2$ (by axiom D1)

$\Gamma_A \vdash (\delta_1 \textit{ is } \delta_2) \textit{ is true}$ (by axiom S12)

$\{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is } (\delta_1 \textit{ is } \delta_2))\} \subseteq \{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is true})\}$

(by transitivity)

$\{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is true})\} = \{|\textit{true}|\} = \mathcal{V}[\![\delta_1 \textit{ is } \delta_2]\!]$ (by definition of $\mathcal{V}$)

The proof is similar under the hypothesis that $\neg\, \mathcal{V}[\![\delta_1]\!] \subseteq \mathcal{V}[\![\delta_2]\!]$.

7. $\sigma \equiv (\sigma_1 \vee \sigma_2)$.

Assume first that: $\Gamma_A \models_{\mathcal{M}_b} \sigma_1$ or $\Gamma_A \models_{\mathcal{M}_b} \sigma_2$

$\Gamma_A \vdash \sigma_1$ or $\Gamma_A \vdash \sigma_2$ (by ind. hyp. and lemma 15)

$\Gamma_A \vdash (\sigma_1 \vee \sigma_2)$ (by completeness of Γ_A)

$\Gamma_A \vdash (\sigma_1 \vee \sigma_2) \textit{ is true}$ (by axiom S12)

$\{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is } (\sigma_1 \vee \sigma_2))\} \subseteq \{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is true})\}$

(by transitivity)

$\{|\delta| \mid \delta \in \text{Ind}, \Gamma_A \vdash (\delta \textit{ is true})\} = \{|\textit{true}|\} = \mathcal{V}[\![\sigma_1 \vee \sigma_2]\!]$ (by definition of $\mathcal{V}$)

The proof is similar under the hypothesis that $\Gamma_A \models_{\mathcal{M}_b} \sigma_1$ and $\Gamma_A \models_{\mathcal{M}_b} \sigma_2$.

We leave out the rest of the proof as it is similar.

Once lemma 16 has been established we can prove the main lemma. In fact from lemma 15 we can deduce the following:

If $\forall\, \sigma\, .\, \mathcal{V}[\![\sigma]\!] = \{|\delta'| \mid \delta' \in \text{Ind}, \Gamma_A \vdash (\delta' \textit{ is } \sigma)\}$ then $\forall\, \sigma\, .\, \Gamma_A \vdash \sigma \Leftrightarrow \Gamma_A \models_{\mathcal{M}_b} \sigma$

The main lemma follows from this and lemma 16.

7. Consistency

The consistency of Omega can be established by means of the following result:

Theorem 17: If an Omega theory Γ has a model, then it is consistent.

Proof: Suppose Γ has a model M, then $\Gamma \not\models_M \textit{false}$. From the completeness result it follows that $\Gamma \not\vdash_M \textit{false}$ which proves the consistency of Γ.

8. Non Primitive Attributions

In this section we will introduce three kind of attributions formally defining them in terms of the primitive attribution *of*.

8.1. Independent Attributions

Each ***with*** attribution is an independent relation from all other attributions in an instance description.

A ***with*** attribution represents a binary relation between attributions and the objects of the description. As such, the ***with*** kind of attribution can be defined as a special case of the ***of*** attributions, as stated by the following definition:

Definition 18:

$(a\ c\ \alpha_1\ (with\ a_1\ \delta)\ \alpha_2)\ same\ (a\ c\ (of\ a_1\ \delta))\ and\ (a\ c\ \alpha_1\ \alpha_2)$

This definition enables us to isolate an attribution in an instance description. In this way that attribution is turned into a binary relation. This binary relation cannot normally be further merged into an n-ary relation, as the following examples illustrate:

(*a* Product (*of* factor$_1$ 2) (*of* factor$_2$ 3)) *is* 6

(*a* Product (*with* factor$_1$ 2) (*with* factor$_2$ 3)) *same*
(*a* Product (*of* factor$_1$ 2)) *and* (*a* Product (*of* factor$_2$ 3)) *same*
(*a* Product (*of* factor$_1$ 6))

As a direct consequence of the definition, ***with*** attributions have the following property of merging:

$((a\ c\ (with\ a_1\ \delta))\ and\ (a\ c\ \alpha))\ is\ (a\ c\ (with\ a_1\ \delta)\ \alpha)$ **(*Merging*)**

However it is easy to verify (by counter example) that merging does not hold for ***of*** attributions.

8.2. Constrained attributions

Sometimes one wants to express constraints on the value of an attribute or describe properties that each value of an attribute must satisfy. For example we use ***with every*** when we want to describe a person who only has male children or specify the type of arguments of a function. For this purpose we introduce the notation ***with every***. For example:

(*a* SquareRoot) *is* (*a* SquareRoot (*with every* arg (*a* PositiveReal)))

expresses the fact that each argument to the square root function must be a positive real number. Another example is:

(*a* Person (*with every* child (*a* Male)))

which describes persons whose children are all male.

Formally we can introduce ***with every*** attributions with the following definition:

Definition 19:

$(a\ c\ (with\ every\ a_1\ \delta_1)\ \alpha)\ same$
$(a\ c\ (with\ a_1\ \delta_1)\ \alpha)\ and$
$(Any\ v\ such\ that\ \forall\ v_1\ .\ \text{Individual}[v_1] \wedge v\ is\ (a\ c\ (with\ a_1\ v_1)) \Rightarrow (v_1\ is\ \delta_1))$

For the ***with every*** kind of attributions the following properties hold:

Theorem 20: (*Fusing*)

$(a\ c\ (with\ every\ a_1\ \delta_1)\ (with\ every\ a_1\ \delta_2))\ same\ (a\ c\ (with\ every\ a_1\ (\delta_1\ and\ \delta_2)))$

$(a\ c\ (with\ every\ a_1\ \delta_1)\ (with\ a_1\ \delta_2))\ is\ (a\ c\ (with\ a_1\ (\delta_1\ and\ \delta_2)))$

$(a\ c\ (with\ every\ a_1\ \delta_1)\ (of\ a_1\ \delta_2)\ \alpha)\ is\ (a\ c\ (of\ a_1\ (\delta_1\ and\ \delta_2))\ \alpha)$

These axioms allow to fuse descriptions in a ***with every*** attribution with any other attribute for the same attribute name. The type of attribution that participates in the fusion becomes the type of the resulting attribution.

The following theorem is needed to establish the properties of *weak fusing.*

Theorem 21:

$(a\ c\ (of\ a_1\ \delta_1))\ same$
$(Any\ v\ such\ that\ \neg\ \forall\ v_1\ .\ \neg(\text{Individual}[v_1] \wedge (v_1\ is\ \delta_1) \wedge (v\ is\ (a\ c\ (of\ a_1\ v_1)))))$

We could prove this theorem by a reasoning analogous to that used in the proof of lemma 10. A simpler proof can given by exploiting the completeness result, i.e. by showing that the statement is true in all interpretations $\mathcal{A} = \langle I, D, R, \mathfrak{I}, \mathfrak{C} \rangle$.

It can be shown that the ***with every*** kind of attribution is subject to all the other axioms given above for attributions.

8.3. Projective attributions

Another non primitive kind of attribution that is often useful is ***with unique.*** The ***with*** kind of attributions imply the existence but not the uniqueness of an individual being the value of the attribute. ***with*** attributions can be merged but cannot be fused. We introduce ***with unique*** as the kind of attribution to use when there is just one individual as value for that attribute. For example:

(*a* Person (*with unique* father (*a* Doctor)))

A version of *fusing*, called *unique fusing*, holds for ***with unique*** attributions. For example:

(*a* Person (*with unique* father (*a* Doctor)) (*with* father Paul))
is (*a* Person (*with unique* father ((*a* Doctor) *and* Paul)))

Note that unless this description reduces to ***Nothing***, we can conclude that **Paul *is* (*a* Doctor)** (lemma 5.9). The kind of attribution ***with unique*** is not primitive and can be defined in terms of other constructs as follows:

Definition 22:

$(a\ c\ (with\ unique\ a_1\ \delta_1)\ \alpha)\ same$
$(a\ c\ (with\ a_1\ \delta_1)\ \alpha)\ and$
$(Any\ v\ such\ that\ \forall\ \delta_2, \delta_3\ .\ \text{Individual}[\delta_2] \wedge \text{Individual}[\delta_3] \wedge$
$(v\ is\ ((a\ c\ (with\ a_1\ \delta_2))\ and\ (a\ c\ (with\ a_1\ \delta_3)))) \Rightarrow (\delta_2\ same\ \delta_3) \wedge (\delta_2\ is\ \delta_1))$

The property of *unique fusing* can be formulated as follows:

Theorem 23: (*Fusing*)

$(a\ c\ (with\ unique\ a_1\ \delta_1)\ (of\ a_1\ \delta_2)\ \alpha)\ is\ (a\ c\ (with\ unique\ a_1\ (\delta_1\ and\ \delta_2))\ \alpha)$

Note that the difference between ***with every*** and ***with unique*** also reflects in a different formulation of *Fusing*; for example:

(*a* Driver (*with unique* car (*a* Datsun)) (*with* car (*a* Blue-car))) *is*
(*a* Driver (*with unique* car ((*a* Datsun) *and* (*a* Blue-car))))

(*a* Driver (*with every* car (*an* American-car)) (*with* car (*a* Ford))) *is*
(*a* Driver (*with* car ((*an* American-car) *and* (*a* Ford))))

8.4. Relationship between WithEvery, WithUnique, With and Of

Let us summarize the properties of the four kinds of attributions that we have discussed.

with unique	***with every***	***with***	***of***
Commutativity	*Commutativity*	*Commutativity*	*Commutativity*
Omission	*Omission*	*Omission*	*Omission*
Monotonicity	*Monotonicity*	*Monotonicity*	*Monotonicity*
Strictness	*Strictness*	*Strictness*	*Strictness*
Merging	*Merging*	*Merging*	
Unique Fusing	*Weak Fusing*		

One would expect that changing ***with every*** or ***with unique*** into ***with*** would give a more general description, and similarly changing ***with*** into ***of***. In fact it can be shown that:

Theorem 24:

1. **(*a* c (*with unique* a_1 δ_1) α) *is* (*a* c (*with every* a_1 δ_1) α)**

2. **(*a* c (*with every* a_1 δ_1) α) *is* (*a* c (*with* a_1 δ_1) α)**

3. **(*a* c (*with* a_1 δ_1)) *is* (*a* c (*of* a_1 δ_1))**

However the following does not hold, when α is not empty:

(*a* c (*with* a_1 δ_1) α) *is* (*a* c (*of* a_1 δ_1) α)

For example:

(*a* Product (*with* a_1 2) (*of* a_2 3)) *same* (*a* Product (*of* a_1 2)) *and* (*a* Product (*of* a_2 3))

and it is not the case that:

((*a* Product (*of* a_1 2)) *and* (*a* Product (*of* a_2 3))) *is* (*a* Product (*of* a_1 2) (*of* a_2 3))

8.5. Data Dependencies

The literature on data bases has investigated the issue of dependencies among data represented in a data base. When such dependencies exist among data, care must be placed when modifying some portion of the data in order to avoid invalidating or destroing meaningful information. Dependencies in fact usually convey semantic or integrity constraints. It is therefore useful to be able to express where such dependencies occur and possibily separate them one from another. Normal forms of data base have been developed as proper

structuring of data which avoid such conflicts.

One kind of data dependencies that have been examined in the literature are functional dependencies. In Omega, we can express that an attribute is not functionally dependent on any other ones, by using the ***with*** notation, as the following result shows:

Theorem 25:

For all $\delta_1, \delta_2, \ldots, \delta_n$
(*a* c (*of* a_1 δ_1) ... (*of* a_n δ_n)) *same* (*a* c (*of* a_1 δ_1)) *and* (*a* c (*of* a_2 δ_2) ... (*of* a_n δ_n))
if and only if
(*a* c (*of* a_1 δ_1) ... (*of* a_n δ_n)) *same* (*a* c (*with* a_1 δ_1) (*of* a_2 δ_2) ... (*of* a_n δ_n))

The interpretation of the above statement is that when the attribute a_1 does not depend on the other attributes $a_2, \ldots, a_n$ then *with* may be used as well for a_1.

For example:

(*a* Complex (*of* real 3) (*of* imag 5)) *same* (*a* Complex (*with* real 3) (*of* imag 5))

In fact:

$\forall\ \delta_1\ \delta_2$.
(*a* Complex (*of* real δ_1)(*of* imag δ_2)) *same*
(*a* Complex (*of* real δ_1)) *and* (*a* Complex (*of* imag δ_2))

The real and imaginary part for a complex number uniquely depend on the complex number being described, in the same way as, given a point in the plane, its cartesian coordinates are uniquely determined.

9. Omega and other Formal Logic Theories

Omega is a more powerful formalism then First Order Predicate Logic, since it allows variables ranging over descriptions (i.e. classes). This provides enough power to express for instance Peano arithmetic with a finite number of axioms. Since there is no layering of descriptions as there is in the hierarchy of sets, Omega is as general as it can be in this respect.

The version of Omega presented here is a first order theory. We are investigating extending the axiomatization to higher orders. In [11] we presented examples of the use of higher order capabilities.

Omega is a Set Theory. However Omega relies on constructors for building new descriptions. In set theory, pairs for instance are built by means of set formation alone. In Omega a Pair constructor can be used to describe pairs of objects. A Pair of two individuals will be an individual itself, therefore separating the inheritance relationship from the component relationship. Omega is a constructive set theory and has no axiom corresponding to the powerset axiom of classical set theory.

10. Language and Metalanguage

The distinction between language and metalanguage has been often overlooked in the literature on

knowledge representation.

We believe that issues largely debated like the distinction between a description and its referent, mention and use of a concept, quotation, opaque operators, belief structures can be resolved if the two levels are clearly understood.

Consider for instance the issue of the distinction between mention and use. The following two statements, both involving the word "Multisyllabe", are respectively examples of mention and use of a concept:

"Multisyllabe" *is* (*a* Multisyllabe)

Multisyllabe *is* (*a* WordProperty)

The relation between the uses of "Multisyllabe" in these sentences is apparent if we move at the metalevel, where the subject of the second sentence (Multisyllabe) can be described as:

(*an* Individual-constant (*with* name "Multisyllabe"))

In Omega there is also a further way of using the same word, as in:

(*a* Multisyllabe) *is* (*a* Word)

The context of quotations and the context of believes or knowledge are instances of opaque contexts. When we cite a statement, like in:

Cantor's theorem says that the power set of natural numbers is a non numerable set

we don't refer to the value of the statement. In fact we don't want to say that Cantor's theorem is true, which is the value of the statement

The power set of natural numbers is a non numerable set

We really intend to express the form of the statement, i.e. we want to give a metadescription of the statement.

The situations in the case of believes is very similar. In

Pat knows that 323-7817 is a phone-number of subscriber Mike

a metadescription for Mike's telephone number is involved, not Mike's telephone number itself. Suppose we denote such metadescription as **'(*a* Phone-number (of subscriber Mike))**. Such metadescription refers to the description **(*a* Phone-number (of subscriber Mike))**, which in turns refers to 323-7817. Even if it is the case that **(*a* Phone-number (of subscriber Mike)) *same* (*a* Phone-number (of subscriber Fred))**, we cannot substitute the metadescription **'(*a* Phone-number (of subscriber Fred))** for **'(*a* Phone-number (of subscriber Mike))** in the above sentence, since these two descriptions are definitely not ***same***. So it is not possible to derive the paradoxical conclusion that Pat knows Fred's phone number.

McCarthy presents a quite similar solution to this problem [19], without connecting his notion of "concepts as objects" to metadescriptions.

A detailed discussion of the issue and uses of metadescriptions remains outside the scope of this paper.

11. Conclusion

Many interesting ideas about knowledge representation have been obscured by the lack of a comprehensive formalism. We have discussed the logic Omega as a proposal for filling this gap. Omega provides most of the attractive features of the formalisms that have been used in representing knowledge: predicate logic, semantic networks, set theory, relational calculus. These features are combined in a simple and unified description system. The results of this paper establish Omega as a solid basis on which to build a theory of knowledge representation.

Inheritance and attributions are two major structuring mechanism in Omega. Though inheritance is used in other formalisms as well, the axiomatization of Omega gives a precise account of the semantics and the properties of such concept.

The semantics presented in the paper also allows to address and clarify a number of issues related to the use of attributions: interaction between attributions, merging and inheritance; functional dependencies among attributions; different kinds of attributions for different purposes: ***with*** attributions to express *part-of* relations, ***of*** attributions to express *arguments-value* relations.

Despite the richness of the system, Omega is still a simple system, where the number of primitive concepts is quite small and the axiomatization quite compact.

12. Acknowledgments

Carl Hewitt has been the leading force in the development of Omega. He proposed to provide an axiomatization for Omega and developed with us the first formulation of the axioms [11]. He contributed the idea of separating ***with*** and ***of*** attributions and introduced the concept of ***with every*** attributions. He also suggested the title for this paper. We are greatly indebted to Albert Meyer and Yuri Gurevich for pointing out that the class of models we had originally provided for Omega was too restricted. Luca Cardelli and Giuseppe Longo thoroughly discussed our ideas in the early stages of this work. William Clinger has provided constructive criticism and encouragement. Mike Brady has given several useful suggestions and comments. Charles Brown, William Kornfeld, David McAllester and Renzo Orsini have suggested improvements to the presentation.

References

1. Attardi, G., Barber, G. and Simi, M. "Towards an Integrated Office Work Station". *Strumentazione e Automazione* (March 1980).

2. Bobrow, D. G. and Winograd, T. "An Overview of KRL, a Knowledge Representation Language". *Cognitive Science 1*, 1 (1977).

3. Brachman, R.J., Ciccarelli, E., Greenfeld, N. and Yonke, M. "KLONE Reference Manual". Report 3848, BBN, 1978.

4. Brachman, R. J. "A Structural Paradigm for Representing Knowledge". Report 3605, Bolt Beranek and Newman Inc., May, 1978.

5. Burstall, R. and Goguen, J. "The Semantics of Clear, a Specification Language". In *Lecture Notes in Computer Science*, Spring Verlag, 1980. Proc. of Advanced Course on Abstract Software Specifications

6. de Kleer, J., Doyle, J., Rich, C., Steele, G. L. and Sussman, G. J. AMORD: a Deductive Procedure System. AI-Memo 435, MIT, January, 1978.

7. De Jong, S.P. and Byrd, R.J. "Intelligent Form Creation in the System for Business Automation (SBA)". Research Report RC 8529, IBM T.J. Watson Research Center, 1980.

8. Deliyanni, A., Kowalski, R.A. "Logic and Semantic Networks". *Comm. ACM 22*, 3 (1979), 184-192.

9. Fahlman, S. *"NETL: A System for Representing and Using Real-World Knowledge"*. MIT Press, 1979.

10. Goguen, J. and Tardo, J. "An Introduction to OBJ: a Language for Writing and Testing Algebraic Specifications". Proc. of Conference on Specifications of Reliable Software, IEEE Computer Society, 1979, pp. 170-189.

11. Hewitt, C., Attardi, G. and Simi, M. "Knowledge Embedding with the Description System OMEGA". Proc. of First National Annual Conference on Artificial Intelligence, Stanford University, August, 1980, pp. 157-163.

12. Ichbiah, J. D. et al. *Reference Manual for the ADA Programming Language.* Honeywell Systems, 1980.

13. Kalish and Montague. *"Logic: Techniques of Formal Reasoning"*. Harcourt, Brace and World, 1964.

14. Kornfeld, W. A. "Using Parallel Processing for Problem Solving". AI-Memo 561, MIT, December, 1979.

15. Kornfeld, W.A. and C. Hewitt. "The Scientific Community Metaphore". *IEEE Systems Man & Cybernetics 11*, 1 (January 1981).

16. Liskov, B. et al. *CLU Reference Manual.* MIT/LCS, 1979. TR-225.

17. Martin, R. M. "A homogeneous system for formal logic". *The Journal of Symbolic Logic 8*, 1 (1943).

18. Martin, W. "Descriptions and Specialization of Concepts". In Patrick H. Winston and Richard H. Brown, Ed., *Artificial Intelligence: An MIT Perspective*, MIT Press, Cambridge, Massachusetts, 1979.

19. McCarthy, John. Epistemological Problems of Artificial Intelligence. Proc. of Fifth International Joint Conference on Artificial Intelligence, IJCAI, 1977, pp. 1038-1044.

20. Mylopoulos, J. and Wong, H. "Some Features of the Taxis Data Model". Proc. of the Conference on Very Large Data Bases, IEEE Computer Society, 1980, pp. 399-410.

21. Russell, B. "On Denoting". *Mind 14* (1905), 479-493.

22. Scott, D. "Existence and Description in Formal Logic". In *Bertrand Russell: Philosopher of the Century,* R. Schoenman, Ed., G. Allen & Unwin Ltd., London, 1969.

23. Shapiro, S.C. "Review of NETL: A System for Representing and Using Real-World Knowledge, by S.E. Fahlman". *American Journal of Computational Linguistics 6*, 3-4 (July-December 1980).

24. Shoenfield, J. R. *"Mathematical Logic"*. Addison Wesley, 1967.

25. Steels, L. "Reasoning Modeled as a Society of Communicating Experts". AI Lab Technical Report 542, MIT, June, 1979.

26. Wirth, N. "The Programming Language PASCAL". *Acta Informatica* , I (1971), 35-63.