# Outline of a Calculus of Type Subsumption

Hassan Ait-Kaci

Technical Report MS-CIS-83-34

Department of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania

August 1983

## Table of Contents

## List of Figures

## 1. Abstract

This paper is a brief analysis of the notion of syntactic representation of types followed by a proposal of a formal calculus of type subsumption. The idea which is developed centers on the concept of *indexed term*, an extension of the definition of algebraic terms relaxing the *fixed arity* and *fixed indexing* constraints, and which allows term symbols to have some pre-order structure. It is shown that the structure on the set of symbols can be "homomorphically" extended to indexed terms to what is defined to be a *subsumption ordering*. Furthermore, when symbols have a lattice structure, this structure extends to a lattice of indexed terms. The notions of *unification* and *generalization* are also shown to fit the extension, and constitute the *meet* and *join* operations.

## 2. Informal Discussion

The approach which is developped in this paper deals with the notion of representation and classification of objects. I shall assume some elementary background in universal algebra, formal logic and automatic theorem proving.

### 2.1. Thesis

*Subtyping* is concerned with capturing the notion of *subsumption* among abstract objects. Thus, I would like to define a notational system for representing *approximations* of objects of which one conceives in one's mind. Moreover, I want this system to contain some mechanism which could automatically classify object thus represented objects in a fashion which is congruent with their interpretation as approximations.

An example of such a system is provided by first-order *terms* or *trees* in universal algebra and logic. For example, I would like to express the fact that a person has a name, a birth date, and a sex. Representing a thus specified generic person as a term could be `person(x,y,z)`. Then, by a convention remembered at interpretation, the symbol `person` at the root of a term denotes a person object, and the variables `x`, `y`, `z` as *place markers* for a person's name, date of birth, and sex, respectively. The classification mechanism in this model is *term instantiation*, or *one-way* first-order unification.[1] The meaning of variables is that they stand for incomplete information and may be substituted for by terms. Thus, `person(Hassan,y,z)` denotes any person named Hassan, and `person(Hassan,date(14,June,y),z)` designates any person named Hassan and born the $14^{th}$ of June. The term appearing as the date of birth in the latter `person` illustrates the substitution process. If I choose to define a type to be a first-order term as shown, and the type classification ordering to be term instantiation, then I have at hand a type system as wished. Indeed, the types thus defined form a *lattice* whose *meet* operation (*i.e.*, greatest lower bound) is first-order unification [5], and whose *join* operation (*i.e.*, least upper bound) is first-order *anti-unification*, or *generalization* [4]. In fact, a calculus for the representation of semantic networks was proposed in [2], based on such a lattice as above.

---

[1] That is, based on the partial ordering of terms defined as: $t_1 \preceq t_2$ if and only if there exists a substitution $\theta$ such that $t_1 = t_2\theta$.

**2.2. Antithesis**

There is however a certain amount of inflexibility inherent to the definition of types as terms. Firstly, a term is a finitely branching tree. In particular, it has a *fixed number of arguments*. If I want to extend the definition of a person to also have a marital status, I must entirely redefine the type `person` to take one more argument, and hence revise all previously used instances of a person. Secondly, a term has a *fixed order of arguments*. This is very convenient to consistently interpret position within a term as having a fixed meaning. For example, in a `person` term, the first argument is once and for all meant to denote the person's name. Indeed, this is also taken advantage of by the unification process; *i.e.*, in order to match, two terms are expected to have their corresponding subterms in the same order. This is the same principle used in most programming languages to pass procedure parameters. As a result, one must constantly keep in mind the original intended interpretation of the order of subterms. Thirdly, type subsumption as one-way pattern-matching is forcing a *common syntactical pattern* for all terms in a chain in the lattice. For example, if I define a type `student(x,y)` where `(x,y)` stand, in this order, for school and subject, then I cannot express that I also intend a student to be a person, since a type is identified by its constant root symbol and `student` is distinct from `person`. Finally, there is no provision in the definition of a term for specifying any *restriction on the pattern* of subterms. For example, restricting the name of a person to terms whose root symbols belong to, or better yet do *not* belong to a given set, is not syntactically possible.

The foregoing shortcomings of the first-order term model of types make it rather look rather limited. However, it has appeal because of its solid formal grounds, its simplicity, and its use as the basic model of types of such clear and clean programming languages as PROLOG.[2] It would be of great advantage if this model of types could be enhanced so that it may keep its elegance and sound formal basis, lend itself to a powerful interpretation scheme, and yet overcome the limitations explicated above. The remainder of this section is the description of such a data model.

---

[2] I am not referring to any particular implementation, but rather to the abstract and pure language

**2.3. Synthesis**

I propose to modify the notion of a type by extrapolating on the classical definition of a term. Let's first relax the fixed-arity constraint; *i.e.*, a term may have an unbounded number of subterms. Next, let's relax the fixed-position constraint by *explicitly* indexing or labeling the arguments. The reader familiar with ADA will note that this language allows a procedure call's actual parameters to be specified either by position, or possibly out of order by explicit labeling. However, in ADA *all* actual parameters *must* be present. In our case, since a type can now have a *potentially infinite* number of attributes, all that is ever needed is to specify only those which are relevant at any given time. For example, `person(name:Hassan)` denotes all persons named Hassan, and `person(sex:Male)` stands for all male persons. Furthermore, let's assume some partial ordering on the root symbols. This can easily be extended to an ordering on terms in a way very similar to a homomorphic extension. For example, if the symbols `person` and `student` are such that `student`$\prec$`person`, then I can consistently say that `student(name:Hassan,subject:CIS)` is a subtype of `person(name:Hassan, subject:CIS)`.

I shall call this kind of extension of a term an *indexed term*, or more figuratively a *type template*. Its definition stays inductive in essence as is the definition of a term in universal algebra. The idea is based on the concept of *multi-sorted* terms with the very peculiar difference that the set of sorts *is* the set of terms! This is quite a new formal window through which to look at data and program structures that makes them syntactically undistinguishable. This forces re-thinking of many related notions. The concepts of variable and symbol which are central in programming as well as formal languages are to be construed in a completely different yet more general way. Indeed, I shall try to explain that if symbols are partially ordered, the notion of variable is but the restrictive designation of a *maximum element*. Symbols, and indexed terms for that matter, may be specified as *upper bound constraints* within other terms. I shall try and show how a natural extension of a partial ordering on the symbols may be consistently defined on indexed terms. Such classical operations as *variable substitution*, *term unification*, *etc.*, take on a new interpretation of which the classical well-known notions are particular cases.

The next section describes the grounds on which the order structure of the set of indexed terms will be resting.

## 3. A Formal Calculus

### 3.1. On Indexed Terms

I shall call an *index set* any countable, not necessarily finite, totally ordered set. I shall use the symbol "$<$" (resp., "$\leq$") to denote an index set's ordering (resp., its reflexive closure). Examples of index sets are the set of natural numbers naturally ordered, and the set of all strings on a linearly ordered alphabet ordered lexicographically. An *indexing function* is a strictly increasing function from an initial segment of $\mathbf{N}_+$, the set of positive integers, to an index set $I$. I shall refer to such a function as an *I-indexing*. The notation $[n]$ stands for the initial segment $\{1,\ldots,n\}$; and $[0]$, for $\emptyset$.

> **Definition 1:** Let $S$ be a set of *symbols*; and $\mathcal{V}$ a set of *variables*. Let $I$ an index set. The set $\mathcal{T}_{I S \mathcal{V}}$ of *indexed terms* (or *indexed trees*) is the set inductively generated as follows:
>
> - Variables are indexed terms;
> - Symbols are indexed terms;
> - Otherwise, an index term is a triple of the form $(f,\iota,T)$, where f is a symbol in $S$, $\iota$ is an $I$-indexing (*i.e.*, $\iota:\mathbf{N}_+\rightarrow I$), and $T = \{t_{\iota(1)},\ldots,t_{\iota(n)}\}$ is a set of indexed terms.

One readily recognizes a straightforward extension of the notion of first-order terms in definition 1. Indeed, the classical definition corresponds to the case where $I$ is $\mathbf{N}_+$, and all indexing functions are the identity on $\mathbf{N}_+$. The set of variables of an indexed term $t$ will be denoted as $\mathit{Var}(t)$.

The introduction of indexing is not merely to make a complicated matter of a simple one, but to allow for more flexibility of representation for terms along the lines of what was discussed in section 2.3, page 4. This will also allow a more general treatment of the lattice-theoretic properties of subsumption which will serve our eventual calculus of partially ordered types. For now, one can always rely on the classical term

representation of the form $f(t_1,\ldots,t_n)$ to ground one's intuition of what is to come. Indeed, as mentioned before, this representation is but a particular syntactic variation of the indexed term $(f,\iota,\{t_{\iota(1)},\ldots,t_{\iota(n)}\})$, where $\iota$ is the identity on $N_+$. I shall often use this classical notation in illustrative examples which would otherwise be notationally cumbersome if given for general indexed terms.

Next, we need a similar generalization of the concept of *variable substitution*. As usual, a *substitution* is a function from $\mathcal{V}$ to $\mathcal{T}_{IS\mathcal{V}}$ which is the identity function almost everywhere, except for a finite set of variables. A substitution $\theta$ will thus be identified to a finite set of pairs $\{(v_1,t_1),\ldots,(v_n,t_n)\}$ such that $v\theta=t_i$ if $v=v_i$, and $v\theta=v$ otherwise. This is extended to an indexed term subsitution $\theta^*$ in a "homomorphic" way as follows:

- $t\theta^* = t\theta$ if $t$ is a variable;
- $t\theta^* = t$ if $t$ is a symbol;
- $(f,\iota,\{t_{\iota(1)},\ldots,t_{\iota(n)}\})\theta^* = (f,\iota,\{t_{\iota(1)}\theta^*,\ldots,t_{\iota(n)}\theta^*\})$.

In general, the "homomorphic" extension will be confused for the variable substitution itself, and therefore the notation "$t\theta$" will be used when it is actually meant "$t\theta^*$".

The composition of substitutions is defined exactly as usual, and I recall it next for the sake of completeness.

> **Definition 2:** Let $\theta = \{(v_i,s_i)\}_{i=1}^m$ and $\phi = \{(w_j,t_j)\}_{j=1}^n$ two substitutions. The *composition* of $\theta$ and $\phi$ is the substitution denoted $\theta\phi$, and defined as $\theta\phi = (\{(v_i,s_i\phi)\}_{i=1}^m \cup \phi) - \{(v,v) \mid v\in\{v_i\}_{i=1}^m\} - \{(w_j,t_j) \mid j\in[n]$, and $\exists i\in[m]$, $v_i=w_j\}$.

Applying the composition $\theta\phi$ to a term is then the same as first applying $\theta$, then applying $\phi$. Let's also recall that substitution composition is associative, non-commutative, and that it defines a partial ordering on the set of substitutions: $\theta$ is *less* than $\phi$ if there exists a substitution $\psi$ such that $\phi=\theta\psi$.

When an equivalence relation is defined on terms, particular substitutions relatively to given terms, called *unifiers* and *generalizers*, are defined as follows.

**Definition 3:** Let $\equiv$ be an equivalence relation on the set of terms. Let $s$ and $t$ be two terms. A *unifying substitution*, or *unifier*, for $s$ and $t$ is a substitution $\theta$ such that $s\theta \equiv t\theta$. A *generalizing pair of substitutions*, or *pair of generalizers*, for $s$ and $t$ is a pair of substitutions $(\theta, \phi)$ such that there is a term $u$, and $u\theta \equiv s$ and $u\phi \equiv t$.

For example, using the classical algebraic term representation and term equality as an equivalence relation on terms, consider the terms $s = f(x,g(a))$ and $t = f(h(b),y)$. Then, $\{(x,h(b)),(y,g(a))\}$ is a unifier for $s$ and $t$, and $(\{(y,g(a))\},\{(x,h(b))\})$ is a pair of generalizers for $s$ and $t$ with generalized term $f(x,y)$.

### 3.2. On Subsumption

A *(partial) ordering* on a set S is a binary relation on S which is reflexive, anti-symmetric, and transitive. A *quasi-ordering* on a set S is a binary relation on S which is reflexive and transitive. A quasi-ordering $\prec$ on S induces an equivalence relation $\sim$ on S defined as $\prec \cap \prec^{-1}$. Furthermore, the quasi-ordering $\prec$ induces an ordering relation $\preceq$ on the quotient set $S/\sim$ defined as follows: $[x] \preceq [y]$ if and only if $x \prec y$.[3]

**Definition 4:** Let $\mathcal{S}$ be a set of symbols quasi-ordered by $\prec$, $\mathcal{V}$ a set of variables, and $I$ an index set. The *subsumption* relation $\preceq$ induced by $\prec$ on the set of indexed terms $\mathcal{T}_{I\mathcal{S}\mathcal{V}}$ is inductively defined as follows; $s \preceq t$ if and only if there exists a substitution of variables $\theta$ such that:

1. *either* $t$ is a variable and $s = t\theta$;

2. *or* $s = (f,\iota,\{s_{\iota(1)},\ldots,s_{\iota(m)}\})$, $t = (g,\eta,\{t_{\eta(1)},\ldots,t_{\eta(n)}\})$,[4] and

   a. $f \prec g$;

   b. $n \leq m$;

   c. there exists a $[m]$-indexing $\kappa: [n] \rightarrow [m]$, such that $\iota(\kappa(i)) = \eta(i)$ and $s_{\iota(\kappa(i))} \preceq t_{\eta(i)}\theta$, $\forall i,\ 1 \leq i \leq n$.

---

[3] This is well-defined since it can be easily verified that it does not depend on particular class representatives.

[4] Here and in all that follows, we use the notational convention that a term of the form $(f,\iota,\{t_{\iota(1)},\ldots,t_{\iota(n)}\})$ with $n=0$ is equivalent to the symbol $f$.

This definition is meant to generalize the *simple* subsumption notion corresponding to the orthodox term instantiation. Indeed, this is what it would reduce to if definition 4 was limited to the case where the quasi-ordering on $\mathcal{S}$ is equality, all indexings are the identity on $\mathbb{N}_+$, and terms have fixed arity.

To illustrate this general notion of subsumption, let's consider the set of symbols $\mathcal{S}=\{\mathtt{a},\mathtt{b},\mathtt{f},\mathtt{g},\mathtt{h}\}$ such that $\mathtt{b}\prec\mathtt{a}$ and $\mathtt{f}\prec\mathtt{g}$. In the following example, let's assume that all indexing is systematically the identity on $\mathbb{N}_+$. According to definition 4, we thus have:

$$\mathtt{f}(x,x,\mathtt{a}) \preceq \mathtt{g}(y,z) \tag{1}$$

$$\mathtt{f}(\mathtt{f}(\mathtt{a},x),\mathtt{g}(y,\mathtt{b}),\mathtt{h}(x)) \preceq \mathtt{g}(\mathtt{g}(z),\mathtt{g}(u,z)) \tag{2}$$

One can verify that a substitution for inequality (1) is $\{(y,x),(z,x)\}$, and one for inequality (2) is $\{(z,\mathtt{a}),(u,y)\}$.

**Theorem 5:** Let $\mathcal{S}$ be a set of symbols quasi-ordered by $\prec$, $\mathcal{V}$ a set of variables, and $I$ an index set. Then, the subsumption relation $\preceq$ induced by $\prec$ on the set of indexed terms $\mathcal{T}_{I\mathcal{S}\mathcal{V}}$ is a quasi-ordering.

**Proof:** We must prove that $\preceq$ is reflexive and transitive. We proceed by structural induction. For reflexivity, we must verify that for any term $\mathtt{t}$ we have $\mathtt{t}\preceq\mathtt{t}$. The base cases are when $\mathtt{t}$ is a variable or a symbol and are trivial. The inductive case for $\mathtt{t} = (\mathtt{f},\iota,\{\mathtt{t}_{\iota(1)},\ldots,\mathtt{t}_{\iota(\mathtt{n})}\})$, with $\mathtt{n}\neq 0$ is also immediate, taking $\theta$ to be the empty substitution, and $\kappa$ the identity on $[\mathtt{n}]$.

As for transitivity, let $\mathtt{s}$, $\mathtt{t}$, $\mathtt{u}$ be terms such that $\mathtt{s}\preceq\mathtt{t}$ and $\mathtt{t}\preceq\mathtt{u}$. If $\mathtt{u}$ is a variable, then immediately $\mathtt{s}\preceq\mathtt{u}$. Now, for the inductive step, let's assume $\mathtt{u}$ is not a variable. Then, by the definition of subsumption, $\mathtt{t}$ is necessarily not one either. And hence, $\mathtt{s}$ must also be a non-variable. Writing $\mathtt{s} = (\mathtt{f},\iota,\{\mathtt{s}_{\iota(1)},\ldots,\mathtt{s}_{\iota(\mathtt{m})}\})$, $\mathtt{t} = (\mathtt{g},\eta,\{\mathtt{t}_{\eta(1)},\ldots,\mathtt{t}_{\eta(\mathtt{n})}\})$, and $\mathtt{u} = (\mathtt{h},\delta,\{\mathtt{u}_{\delta(1)},\ldots,\mathtt{u}_{\delta(\mathtt{p})}\})$, we have $\mathtt{p}\leq\mathtt{m}$, $\mathtt{f}\prec\mathtt{h}$ (since $\mathtt{s}\preceq\mathtt{t}$ and $\mathtt{t}\preceq\mathtt{u}$, and by transitivity of $\leq$ and $\prec$), and there exist two substitutions $\theta_1$, and $\theta_2$, and a $[\mathtt{m}]$-indexing $\kappa_1$, and a $[\mathtt{n}]$-indexing $\kappa_2$ such that:

$$\iota(\kappa_1(\mathtt{i})) = \eta(\mathtt{i}), \quad \forall \mathtt{i}\in[\mathtt{n}] \tag{3}$$

$$\eta(\kappa_2(\mathtt{j})) = \delta(\mathtt{j}), \quad \forall \mathtt{j}\in[\mathtt{p}] \tag{4}$$

$$\mathtt{s}_{\iota(\kappa_1(\mathtt{i}))} \preceq \mathtt{t}_{\eta(\mathtt{i})}\theta_1, \quad \forall \mathtt{i}\in[\mathtt{n}] \tag{5}$$

$$\mathtt{t}_{\eta(\kappa_2(\mathtt{j}))} \preceq \mathtt{u}_{\delta(\mathtt{j})}\theta_2, \quad \forall \mathtt{j}\in[\mathtt{p}] \tag{6}$$

by definition of subsumption. Now, define the substitution $\theta{=}\theta_2\theta_1$, and the $[\mathtt{m}]$-indexing $\kappa{=}\kappa_1\circ\kappa_2$. Hence, using equations (3) and (4) together with associativity of composition, we obtain:

$$\iota(\kappa(\mathtt{j})) = \iota(\kappa_1(\kappa_2(\mathtt{j}))) = \eta(\kappa_2(\mathtt{j})) = \delta(\mathtt{j}), \quad \forall \mathtt{j}\in[\mathtt{p}]$$

and combining this with inequalities (5) and (6), we get:

$$\mathtt{s}_{\iota(\kappa(\mathtt{j}))} \preceq \mathtt{t}_{\eta(\kappa_2(\mathtt{j}))}\theta_1 \preceq \mathtt{u}_{\delta(\mathtt{j})}\theta_2\theta_1 = \mathtt{u}_{\delta(\mathtt{j})}\theta, \quad \forall \mathtt{j}\in[\mathtt{p}].$$

By inductive hypothesis on the transitivity of $\preceq$, this states that we have $\mathtt{s}\preceq\mathtt{u}$, and terminates the proof.

As a corollary, the subsumption quasi-ordering on $\mathcal{T}_{IS\mathcal{V}}$ induces an equivalence relation $\approx$ on $\mathcal{T}_{IS\mathcal{V}}$. Furthermore, it induces a partial ordering on the quotient set $\mathcal{T}_{IS\mathcal{V}}/\approx$, where $\approx$ is defined as $\preceq \cap \preceq^{-1}$, as described in the beginning of this section.

**Lemma 6:** The set of variables $\mathcal{V}$ is an equivalence class modulo $\approx$; furthermore, $\mathcal{V}$ is *the* maximum element in $\mathcal{T}_{IS\mathcal{V}}/\approx$.

The proof of lemma 6 is immediate. We can then justifiably adopt the notation $\top_{\mathcal{T}_{IS\mathcal{V}}/\approx}$, or simply $\top$ when there is no ambiguity, to denote $[\mathtt{v}]$, for any variable $\mathtt{v}$ in $\mathcal{V}$.

It must be clear, at this point, that most notions defined for regular algebraic terms are easily generalizable to indexed terms. Thus, let's define the concept of congruence.

**Definition 7:** Let $\mathcal{T}_{IS\mathcal{V}}$ be defined as before. Let $\sim$ be an equivalence relation on $S$, and $\approx$ an equivalence relation on $\mathcal{T}_{IS\mathcal{V}}$. The relation $\approx$ is an *indexed congruence* (or simply a congruence), if and only if whenever two symbols $\mathtt{f}$ and $\mathtt{g}$ are such that $\mathtt{f}\sim\mathtt{g}$, and $\mathtt{n}$ pairs of indexed terms $\mathtt{s}_\mathtt{i}$, $\mathtt{t}_\mathtt{i}$, $\mathtt{i}\in[\mathtt{n}]$, are such that $\mathtt{s}_\mathtt{i}\approx\mathtt{t}_\mathtt{i}$, then $(\mathtt{f},\iota,\{\mathtt{s}_1,\ldots,\mathtt{s}_\mathtt{n}\}) \approx (\mathtt{g},\iota,\{\mathtt{t}_1,\ldots,\mathtt{t}_\mathtt{n}\})$, for any $I$-indexing $\iota{:}[\mathtt{n}]\rightarrow I$.

It is therefore not difficult to establish the following lemma.

**Lemma 8:** The equivalence relation $\approx \; = \; \preceq \cap \preceq^{-1}$ induced by a subsumption relation $\preceq$ on the set of terms $\mathcal{T}_{IS\mathcal{V}}$ is a congruence relation.

### 3.3. On Indexings

The result that will be of prime interest is stated as theorem 13. In order to be complete, the preliminaries necessary for its proof must contain a study of the particular order structure on the set of $I$-indexings suggested in definition 4. This is what is further described and elaborated upon in this section.

**Definition 9:** Let $I$ be an index set, and let $\iota:[\mathtt{m}]\rightarrow I$ and $\eta:[\mathtt{n}]\rightarrow I$ be two $I$-indexings. Define the binary relation $\ll$ on $I$-indexings as: $\iota \ll \eta$ if and only if (1) $\mathtt{n} \leq \mathtt{m}$, and (2) there exists a $[\mathtt{m}]$-indexing $\kappa:[\mathtt{n}]\rightarrow[\mathtt{m}]$, such that $\eta = \iota \circ \kappa$.

The verification of the following proposition is straightforward.

**Proposition 10:** The relation $\ll$ defined in definition 9 is an ordering relation.

We can define two binary operations $\vee$ and $\wedge$ on the set of $I$-indexings. Let $\iota:[\mathtt{m}]\rightarrow I$ and $\eta:[\mathtt{n}]\rightarrow I$ be two $I$-indexings. Let $\mathtt{p} = |\iota([\mathtt{m}]) \cap \eta([\mathtt{n}])|$, and $\mathtt{q} = |\iota([\mathtt{m}]) \cup \eta([\mathtt{n}])|$. Let $\iota \vee \eta:[\mathtt{p}]\rightarrow I$ and $\iota \wedge \eta:[\mathtt{q}]\rightarrow I$ be the two $I$-indexings defined as:

$$(\iota \vee \eta)(\mathtt{i}) = \text{the } \mathtt{i}^{\text{th}} \text{ least element in } \iota([\mathtt{m}]) \cap \eta([\mathtt{n}]),\ \forall \mathtt{i} \in [\mathtt{p}] \tag{7}$$

$$(\iota \wedge \eta)(\mathtt{i}) = \text{the } \mathtt{i}^{\text{th}} \text{ least element in } \iota([\mathtt{m}]) \cup \eta([\mathtt{n}]),\ \forall \mathtt{i} \in [\mathtt{q}] \tag{8}$$

where "least" refers to the total ordering on $I$.

This leads to the following theorem:

**Theorem 11:** The binary operations defined by expressions (7) and (8) define a lattice structure on the set of $I$-indexings ordered by $\ll$, and yield respectively the LUB and GLB of two $I$-indexings.

**Proof:** Let $\iota$, $\eta$, $\mathtt{n}$, $\mathtt{m}$, $\mathtt{p}$, $\mathtt{q}$ be defined as above. We need to establish first that $\iota \vee \eta$ and $\iota \wedge \eta$ are indeed respectively an upper bound and a lower bound of

both $\iota$ and $\eta$. For this, consider the four indexings $\kappa_1$:[p]$\rightarrow$[m], $\kappa_2$:[p]$\rightarrow$[n], $\kappa_3$:[m]$\rightarrow$[q], and $\kappa_4$:[n]$\rightarrow$[q], respectively defined by expressions (9), (10), (11), and (12).

$$\kappa_1(\mathtt{i}) = \mathtt{j}, \text{ where } \mathtt{j}\in[\mathtt{m}] \text{ and } \iota(\mathtt{j}) = (\iota\vee\eta)(\mathtt{i}),\ \forall \mathtt{i}\in[\mathtt{p}] \tag{9}$$

$$\kappa_2(\mathtt{i}) = \mathtt{j}, \text{ where } \mathtt{j}\in[\mathtt{n}] \text{ and } \eta(\mathtt{j}) = (\iota\vee\eta)(\mathtt{i}),\ \forall \mathtt{i}\in[\mathtt{p}] \tag{10}$$

$$\kappa_3(\mathtt{i}) = \mathtt{j}, \text{ where } \mathtt{j}\in[\mathtt{q}] \text{ and } (\iota\wedge\eta)(\mathtt{j}) = \iota(\mathtt{i}),\ \forall \mathtt{i}\in[\mathtt{m}] \tag{11}$$

$$\kappa_4(\mathtt{i}) = \mathtt{j}, \text{ where } \mathtt{j}\in[\mathtt{q}] \text{ and } (\iota\wedge\eta)(\mathtt{j}) = \eta(\mathtt{i}),\ \forall \mathtt{i}\in[\mathtt{n}] \tag{12}$$

It is easy to verify that these are well-defined four indexings such that, on one hand, $\iota\circ\kappa_1 = \eta\circ\kappa_2 = \iota\vee\eta$, and hence, $\iota\vee\eta$ is an upper bound for both $\iota$ and $\eta$; and on the other hand, $(\iota\wedge\eta)\circ\kappa_3 = \iota$, and $(\iota\wedge\eta)\circ\kappa_4 = \eta$, proving that $\iota\wedge\eta$ is a lower bound for both $\iota$ and $\eta$.

The next thing to prove that $\iota\vee\eta$ is the *least* upper bound. Let $\alpha$:[p']$\rightarrow I$ be a $I$-indexing such that $\iota\ll\alpha$ and $\eta\ll\alpha$. That is $\mathtt{p}'\leq\mathtt{m}$, $\mathtt{p}'\leq\mathtt{n}$, and there exist two indexings $\kappa'_1$:[p']$\rightarrow$[m] and $\kappa'_2$:[p']$\rightarrow$[n] such that $\alpha=\iota\circ\kappa'_1$ and $\alpha=\eta\circ\kappa'_2$. Therefore, $\alpha(\mathtt{i})\in\iota([\mathtt{m}])$ and $\alpha(\mathtt{i})\in\eta([\mathtt{n}])$, $\forall \mathtt{i}\in[\mathtt{p}']$. That is, $\alpha([\mathtt{p}'])\subseteq\iota([\mathtt{m}])\cap\eta([\mathtt{n}])$. But, $\alpha$ is an indexing, and so it is injective. Hence, $\mathtt{p}'\leq\mathtt{p}$. Now, consider the indexing $\beta$:[p']$\rightarrow$[p] defined by $\beta = (\iota\vee\eta)^{-1}\circ\iota\circ\kappa'_1$, where $(\iota\vee\eta)^{-1}$:$(\iota\vee\eta)([\mathtt{p}])\rightarrow$[p] is the inverse function of $\iota\vee\eta$. The function $\beta$ is well-defined and is indeed an indexing. Furthermore, $\iota\vee\eta=\alpha\circ\beta$. Therefore, we have proved that $\alpha\ll\iota\vee\eta$.[5]

Finally, we prove that $\iota\wedge\eta$ is the *greatest* lower bound. Let $\alpha$:[q']$\rightarrow I$ be a $I$-indexing such that $\alpha\ll\iota$ and $\alpha\ll\eta$. That is $\mathtt{m}\leq\mathtt{q}'$, $\mathtt{n}\leq\mathtt{q}'$, and there exist two indexings $\kappa'_3$:[m]$\rightarrow$[q'] and $\kappa'_4$:[n]$\rightarrow$[q'] such that $\iota=\alpha\circ\kappa'_3$ and $\eta=\alpha\circ\kappa'_4$. That is, $\forall \mathtt{i}\in[\mathtt{m}]\cup[\mathtt{n}]$, $\exists \mathtt{j}\in[\mathtt{q}']$, $\alpha(\mathtt{i})=\iota(\mathtt{i})$ or $\alpha(\mathtt{i})=\eta(\mathtt{i})$. We can reformulate this as follows: $\forall \mathtt{k}\in\iota([\mathtt{m}])\cup\eta([\mathtt{n}])$, $\exists \mathtt{j}\in[\mathtt{q}']$, $\alpha(\mathtt{j})=\mathtt{k}$. Again, since $\alpha$ is injective, this proves $\mathtt{q}\leq\mathtt{q}'$. Also, this justifies the sound definition of the indexing $\beta$:[q]$\rightarrow$[q'] as: $\forall \mathtt{i}\in[\mathtt{q}]$, $\beta(\mathtt{i})=\mathtt{j}$, $\mathtt{j}\in[\mathtt{q}']$ and $\alpha(\mathtt{j})=(\iota\wedge\eta)^{-1}(\mathtt{i})$, where $(\iota\wedge\eta)^{-1}$:$(\iota\wedge\eta)([\mathtt{q}])\rightarrow$[q] is the inverse function of $\iota\wedge\eta$. Again, the function $\beta$ is well-defined and is indeed an indexing. Furthermore, $\alpha=(\iota\wedge\eta)\circ\beta$. Therefore, we have proved that $\iota\wedge\eta\ll\alpha$.

---

[5]We could as well have defined $\beta = (\iota\vee\eta)^{-1}\circ\eta\circ\kappa'_2$. The proof works symmetrically.

### 3.4. On Term Lattices

The last step before theorem 13 concerns the definition of algorithms of *unification* and *generalization* for indexed terms.

**Lemma 12:** If the set of symbols $S$ is a lattice, then for two given indexed terms in $\mathcal{T}_{IS\mathcal{V}}$ a *maximum* pair of generalizers always exists and is computable. Also, a *minimum* unifier or proof that no unifier exists is computable.

**Proof:** Given two indexed terms, the algorithm in figure 3-1, page 13, computes their maximum pair of generalizers, and the algorithm in figure 3-2, page 14, computes their minimum unifier or fails. In these two algorithms, it is made implicit use of $\mathbf{p}$, $\mathbf{q}$, $\vee$, $\wedge$, as defined in expressions (7), (8), and of $\kappa_1$, $\kappa_2$, $\kappa_3$, and $\kappa_4$, defined by expressions (9), (10), (11), and (12).

We are finally ready for the main theorem:

**Theorem 13:** If the quasi-ordering $\prec$ on the set of symbols $S$ is a lattice ordering, then the quotient set $\mathcal{T}_{IS\mathcal{V}}/\approx$ augmented with a bottom element $\perp_{\mathcal{T}_{IS\mathcal{V}}/\approx}$, where $\approx$ is defined as $\preceq \cap \preceq^{-1}$, has a lattice structure.

**Proof:** We need to prove the existence of LUB's and GLB's. The expressions *sup* and *inf* in figures 3-3 and 3-4 are well-defined and compute respectively the GLB and the LUB of two congruence classes modulo $\approx$.

## 4. Conclusion

Introducing the notion of indexed term and subsumption as specified in definitions 1 and 4 is aimed at extrapolating and generalizing the essential properties of orderings such as the ones studied in [3] and [4]. The study of this definition of subsumption will be the basis of a semantics of the theory of types which I propose to develop. As an illustration, I intend to use the calculus defined here to formally capture the kind of structures generally proposed as semantic networks. One such example which I think would lend itself to be thus formalized is Kl-One, the knowledge representation language developed by R.Brachman [1]. Another concrete goal is the extension of the logic

```
procedure generalize
   (input
      s,t : indexed term;
      φ1,φ2 : substitution;
   output
      θ1,θ2 : substitution;
      u : indexed term)

   begin
   θ1 := φ1; θ2 := φ2;
   if s is a variable then
      if s = t then u := s
      else
         if s = sφ2 then begin θ2 := θ2∪{(s,t)}; u := s end
         else
            if ∃v∈𝒱 such that vφ1 = s and vφ2 =t then u := v
            else
               begin
               let v∈𝒱 such that v = vφ1 = vφ2;
               θ1 := θ1∪{(v,s)}; θ2 := θ2∪{(v,t)}; u := v
               end
   else
      if t is a variable then
         if t = tφ1 then begin θ1 := θ1∪{(t,s)}; u := t end
         else
            if ∃v∈𝒱 such that vφ1 = s and vφ2 =t then u := v
            else
               begin
               let v∈𝒱 such that v = vφ1 = vφ2;
               θ1 := θ1∪{(v,s)}; θ2 := θ2∪{(v,t)}; u := v
               end
      else
         begin
         let s = (f,ι,{s_ι(1),...,s_ι(m)}), t = (g,η,{t_η(1),...,t_η(n)});
         for i := 1 to p do generalize(s_ι(κ1(i)),t_η(κ2(i)),θ1,θ2,θ1,θ2,u_(ι∨η)(i));
         u := (sup_ζ(f,g),ι∨η,{u_(ι∨η)(1),...,u_(ι∨η)(p)})
         end
   end;
```

**Figure 3-1:** The Generalization Algorithm for Indexed Terms

```
procedure unify
   (input
      s,t : indexed term;
      φ : substitution;
   output
      θ : substitution or fail;
      u : indexed term)

   begin
   θ := φ;
   if s is a variable then
      if t is a variable then
         begin u := t; if s ≠ t then θ := θ{(s,t)} end
      else
         if s∈Var(t) then θ := fail
         else begin u := t; θ := θ{(s,t)} end
   else
      if t is a variable then begin u := s; θ := θ{(t,s)} end
      else
         begin
         let s = (f,ι,{s_ι(1),...,s_ι(m)}), t = (g,η,{t_η(1),...,t_η(n)});
         h := inf_S(f,g);
         if h = ⊥_S then θ := fail
         else
            begin
            i := 1;
            while (i ≤ q) and (θ ≠ fail) do
               begin
               if (∃j, j∈[m] and κ_3(j)=i) and (∃k, k∈[n] and κ_4(k)=i) then
                  unify(s_ι(j)θ, t_η(k)θ, θ, θ, u_(ι∧η)(i))
               else
                  if ∃j, j∈[m] and κ_3(j)=i then u_(ι∧η)(i) := s_ι(j)
                  else
                     if ∃k, k∈[n] and κ_4(k)=i then u_(ι∧η)(i) := t_η(k);
               i := i+1
               end
            if θ ≠ fail then u := (h,ι∧η,{u_(ι∧η)(1),...,u_(ι∧η)(q)})
            end
         end
   end;
```

**Figure 3-2:** The Unification Algorithm for Indexed Terms

```
sup([s],[t]) := [u]
   where generalize(s,t,∅,∅,θ1,θ2,u);
```

**Figure 3-3:** Expression of a LUB in $\mathcal{T}_{ISV}/\approx$

```
inf([s],[t]) := if θ = fail then ⊥T_ISV/≈ else [u]
   where s∈[s] and t∈[t] such that Var(s) ∩ Var(t) = ∅
     and unify(s,t,∅,θ,u);
```

**Figure 3-4:** Expression of a GLB in $\mathcal{T}_{ISV}/\approx$

programming language PROLOG to a *typed* interpreter of indexed term "Horn"-clauses. In fact, this idea introduces a whole study of a definition of *models* for indexed term *algebras*, their existence, and the possibility of extending such notions as algebraic, logic, and denotational semantics. Indeed, the content of this paper has just barely scratched a film off a potentially fecund mathematics. It is an equally interesting challenge to point out either of its positive or negative upshots.

## References

[1] Brachman, R.J.
*A New Paradigm for Representing Knowledge.*
BBN Report 3605, Bolt Beranek and Newman, Cambridge, MA, 1978.

[2] Deliyanni, A., and Kowalski, R.A.
Logic and Semantic Networks.
*Communications of the ACM* 22(3):184-192, 1979.

[3] Plotkin, G.D.
*Lattice Theoretic Properties of Subsumption.*
Memorandum MIP-R-77, Department of Machine Intelligence and Perception, University of Edinburgh, June, 1977.

[4] Reynolds, J.C.
Transformational Systems and the Algebraic Structure of Atomic Formulas.
In D. Michie (editor), *Machine Intelligence 5*, chapter 7. Edinburgh University Press, 1970.

[5] Robinson, J.A.
A Machine-Oriented Logic Based on the Resolution Principle.
*Journal of the ACM* 12(1):23-41, 1965.