DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-9801

A Two-Sorted Interpretation for Annotated Logic

Flávio Soares Corrêa da Silva,
Daniela Vasconcelos Carbogim

fevereiro de 1998

A Two-Sorted Interpretation for Annotated Logic

Flávio Soares Corrêa da Silva, Daniela Vasconcelos Carbogim
Instituto de Matemática e Estatística, Universidade de São Paulo
Cid. Universitária "ASO" – São Paulo (SP) – Brazil
05508-900 – email: fcs@ime.usp.br
Department of Artificial Intelligence, University of Edinburgh
80 South Bridge – Edinburgh, Scotland EH1 1HN – email: danielac@dai.ed.ac.uk

Abstract

Annotated Logic was introduced in [Sub87] as a logic programming language to implement reasoning with uncertainty for knowledge-based systems. Later, it was shown to be adequate to formalise systems in which inconsistencies can be present, without trivialising its reasoning (technically coined "paraconsistent systems" [Cos63]). This Logic was generalised in [BS88], but its foundational aspects were only developed years later [CSV91, CAS91], when annotated propositional and first-order calculi were presented. They are non-classical logics obtained from a positive logic with implication to which is added a weak negation operator.

Recently, Annotated Logic has been advocated to be better suited to reason *about* uncertainty than *with* uncertainty [CdS96]. It has also been proposed as a useful language to reason *about* arguments, rather than to reason *with* (logical) arguments about objective facts [CCdS97]..

In the present article we restate First-order Annotated Logic as a simple two-sorted classical first-order language. Our goal is to expose some relations between this Logic and correspondence (i.e. *relational*) theories of truth [Vis97], thus suggesting the application of this Logic as a flexible tool to build relational theories of uncertainty.

1 Introduction

Annotated Logic was introduced in [Sub87] as a logic programming tool to implement reasoning with uncertainty for knowledge-based systems. The flexibility and practical applicability of this tool was nicely explored and extended in a series of articles [KS89, KS92, KL92, Thi95, NS92, NS90b, NS90a, NS91].

Later, Annotated Logic was shown to be adequate to formalise systems in which inconsistencies can be present, without trivialising its reasoning [BS87] (viz. "paraconsistent formal systems" [Cos63, Cos74]). This Logic was generalised in [BS88], but its foundational aspects were only developed years later [CSV91, CAS91], when annotated propositional and first-order calculi were presented. They are non-classical logics obtained from

a positive logic with implication to which is added a weak negation operator, whose semantics is specified as a function on a complete lattice of truth-values.

Recently, Annotated Logic has been advocated to be better suited to reason *about* uncertainty than *with* uncertainty [CdS96]. It has also been proposed as a useful language to reason *about* arguments, rather than to reason *with* (logical) arguments about objective facts [CCdS97].

In the present article we restate First-order Annotated Logic as a simple two-sorted classical first-order language. Our goal is to expose some relations between this Logic and correspondence (i.e. *relational*) theories of truth [Vis97], thus suggesting the application of this Logic as a flexible tool to build relational theories of uncertainty.

In section 2 we briefly review First-order Annotated Logic as it was presented in [CAS91, Abe92, Car96]. In section 3 we reconstruct this Logic as a two-sorted first-order logic. Finally, in section 4 we present some discussion and our proposed relation between this Logic and a correspondence theory of uncertainty.

2 The First-Order Annotated Predicate Calculus

In this section, we briefly present the First-Order Annotated Calculus $Q\tau$ [CAS91, Abe92, Car96]. This Calculus is based on explicit (i.e. *syntactic*) representations of truth-values, which can represent degrees of (un)certainty about the (two-valued) truth of closed formulae.

Let $|\ \tau\ |$ be a non-empty set of truth-values and $\leq$ an ordering defined over it, such that $\tau = <|\ \tau\ |, \leq>$ is a finite lattice[1]. We denote by $\sqcup$ the least upper bound operator. The elements of τ are called *annotation constants*. In what follows, we define the associated annotated logic programming language $L\tau$.

In τ, we assume the existence of a fixed function $\sim: |\ \tau\ | \rightarrow |\ \tau\ |$. Later, we will see that this function provides the "meaning" of the (weak) negation operator $\neg$ in the system $Q\tau$.

The language $L\tau$ of $Q\tau$ is a first-order language with equality, formed by the following primitive symbols:

1. A countable set of variable symbols, named *individual variables*;
2. Logical connectives $\neg, \wedge, \vee, \rightarrow$;
3. Quantifiers $\forall$, $\exists$;
4. Equality $=$;
5. Auxiliary symbols: parentheses, comma;
6. The elements of τ, named *annotation constants*;
7. For each natural number $n < \omega$, a collection of function symbols of arity n. Function symbols of arity 0 are also called *individual constants*;

[1]The theory is originally defined for complete lattices. In the present article, since we are considering Artificial Intelligence applications, we are committed to languages that can be implemented and have their reasoning procedures effectively computable. Since every finite lattice is complete, we restrain ourselves to the case of finite lattices, thus assuring that every value in $|\ \tau\ |$ can be effectively generated.

8. For each natural number $n < \omega$, a collection of predicate symbols of arity n.

The symbols described in the items 1 to 5 are logical symbols, and they are present in every first-order language with equality. The items 6 to 8 describe the non-logical symbols which characterise $L\tau$.

Terms in $L\tau$ are defined as usual. Formulae are inductively defined as follows:

1. If p is an n-ary predicate symbol, μ is an annotation constant and $t_1, ..., t_n$ are terms of $L\tau$, then $p(t_1, ..., t_n) : \mu$ is called an *annotated atom* or *atomic formula.* Expressions of the form $p(t_1, ..., t_n)$ are called *atoms* of $L\tau$;
2. If t_1, t_2 are terms, then $t_1 = t_2$ is an *atomic formula*;
3. If A is a formula, then $(\neg A)$ is a formula;
4. If A and B are formulae, then $(A \wedge B)$, $(A \vee B)$ and $(A \rightarrow B)$ are formulae;
5. If A is a formula and v is an individual variable, then $(\forall v)A$ and $(\exists v)A$ are formulae.

Intuitively, an annotated atom $p(t_1, ..., t_n) : \mu$ may be interpreted as *the truth-value of* $p(t_1, ..., t_n)$ *is at least* μ. The annotation may also represent a degree of belief (or uncertainty, or reliability) associated to the atom by a reasoning agent. For example, $A : \mu$ can be intuitively interpreted as *it is believed that the truth-value of* A *is at least* μ.

We say that A is a *closed formula* if A does not contain any free occurrence of a variable. We denote by $(\forall)A$ the universal closure of A.

Definition 2.1 (Hyper-literal) *If $p(t_1, ..., t_n) : \mu$ is an annotated atom and k is a natural number, then a hyper-literal is inductively defined as follows:*

1. $\neg^0 p(t_1, ..., t_n) : \mu \equiv p(t_1, ..., t_n) : \mu$;
2. $\neg^k p(t_1, ..., t_n) : \mu = \neg(\neg^{k-1} p(t_1, ..., t_n) : \mu)$, $k \geq 1$.

A formula is called complex if it is not a hyper-literal.

It is worth mentioning that it is not possible to redefine all the logical connectives of the language from only two connectives ($\wedge$ and $\neg$, for example), as in the classical case. This is because the negation works differently for hyper-literals and complex formulae.

In particular, we use additional connectives as abbreviations of certain expressions, such as the bi-conditional ($\leftrightarrow$).

We will now define a semantics for $L\tau$.

Definition 2.2 (Structure) *A structure $\mathcal{A}$ for $L\tau$ is formed by the following objects:*

1. *A non-empty set $|\mathcal{A}|$, called domain or universe. The elements in $|\mathcal{A}|$ are called individuals of $\mathcal{A}$.*
2. *For each function symbol f of $L\tau$, a function $f^{\mathcal{A}} : |\mathcal{A}|^n \rightarrow |\mathcal{A}|$. In particular, for each individual constant c of $L\tau$, $c^{\mathcal{A}}$ is an individual of $\mathcal{A}$.*
3. *For each predicate symbol p of $L\tau$, a function $p^{\mathcal{A}} : |\mathcal{A}|^n \rightarrow |\tau|$.*

Let $\mathcal{V}$ and $\mathcal{T}$ be the sets of variables and terms of $L\tau$, respectively. If $\mathcal{A}$ is a structure for $L\tau$ and $s : \mathcal{V} \rightarrow |\mathcal{A}|$ valuation over the variables for $\mathcal{A}$, then $s : \mathcal{T} \rightarrow |\mathcal{A}|$ is a valuation over the terms obtained as usual.

Definition 2.3 *Let $\mathcal{A}$ be a structure for $L\tau$, s a valuation for $\mathcal{A}$ and A a formula of $L\tau$. Then $\mathcal{A} \models A[s]$ is defined as follows:*

1. *If A is an atomic formula $p(t_1, ..., t_n) : \mu$, then $\mathcal{A} \models A[s]$ iff $p^{\mathcal{A}}(s(t_1), ..., s(t_n)) \geq \mu$.*
2. *If A is a hyper-literal $\neg^k p(t_1, ..., t_n) : \mu, k \geq 1$, then $\mathcal{A} \models A[s]$ iff $\mathcal{A} \models \neg^{k-1} p(t_1, ..., t_n) : \sim \mu$.*
3. *If A is $(\neg F)$, where F is a complex formula, then $\mathcal{A} \models A[s]$ iff it is not the case that $\mathcal{A} \models F[s]$.*
4. *For the other formulae, $\mathcal{A} \models A[s]$ is defined as usual.*

If $\mathcal{A} \models A[s]$, we say that $\mathcal{A}$ *satisfies* A *for* s. We usually write $\mathcal{A} \not\models A[s]$ when it is not the case that $\mathcal{A} \models A[s]$. If $\mathcal{A} \models A[s]$ for every $s : \mathcal{V} \rightarrow |\mathcal{A}|$, we say that $\mathcal{A}$ *is a model of* A and denote by $\mathcal{A} \models A$. A formula A is said to be *logically valid* ($\models A$) if $\mathcal{A} \models A$ for every structure $\mathcal{A}$ for $L\tau$. If Γ is a set of formula of $L\tau$, we say that A is a *semantic consequence* of Γ ($\Gamma \models A$) if, for every structure $\mathcal{A}$ such that $\mathcal{A} \models B$ for every formula $B \in \Gamma$, then $\mathcal{A} \models A$. If $\Gamma = \emptyset$, then $\Gamma \models A$ iff $\models A$.

In this system, concepts such as *free and bounded variables in a formula*, *free occurrence of a variable in a formula* and *substitution of a term for a free variable* are natural extensions of the analogous concepts from the First-Order Classic Predicate Calculus. So, we denote by $A_v[t]$ the formula resulting from substituting the term t in A for the free occurrences of v, when t is replaceable for v in A. When it is clear from the context which variable is being substituted, we write $A[t]$.

In figures 1 and 2 we present, respectively, an axiom system and inference rules for the Predicate Calculus $Q\tau$. In [Abe93, CAS91], we find a proof of the soundness and completeness of this axiomatisation with respect to the semantics proposed above.

Let A, B, C be arbitrary formulae, F, G be complex formulae, p be a predicate symbol, t_i, $1 \leq i \leq n$ be terms, λ, μ, μ_i, $1 \leq i \leq m$ be annotation constants and u, v individual variables of $L\tau$.

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
3. $((A \rightarrow B) \rightarrow A) \rightarrow A$
4. $(A \wedge B) \rightarrow A$
5. $(A \wedge B) \rightarrow B$
6. $A \rightarrow (B \rightarrow (A \wedge B))$

7. $A \rightarrow (A \vee B)$
8. $B \rightarrow (A \vee B)$
9. $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))$
10. $(F \rightarrow G) \rightarrow ((F \rightarrow \neg G) \rightarrow \neg F)$
11. $F \rightarrow (\neg F \rightarrow A)$
12. $F \vee \neg F$
13. $A[t] \rightarrow (\exists v)A(v)$
14. $(\forall v)A(v) \rightarrow A[t]$
15. $p(t_1, ..., t_n) : \perp$
16. $p(t_1, ..., t_n) : \mu \rightarrow p(t_1, ..., t_n) : \lambda,\ \mu \geq \lambda$
17. $(\neg^k p(t_1, ..., t_n) : \mu) \leftrightarrow (\neg^{k-1} p(t_1, ..., t_n) : \sim \mu),\ k \geq 1$
18. $(p(t_1, ..., t_n) : \mu_1 \wedge ... \wedge p(t_1, ..., t_n) : \mu_m) \rightarrow p(t_1, ..., t_n) : \mu,\ \mu = \sqcup_{i=1}^{m} \mu_i$
19. $v = v$
20. $(v = u) \rightarrow (A(v) \leftrightarrow A_v'[u])$,
where $A_v'[u]$ is the formula that results from substituting u for some of the free occurrences of v in A such that u is replaceable for v in A.

Figure 1: Axiom System for the Predicate Calculus $Q\tau$

Inference Rules:

1. $\dfrac{A \quad A \rightarrow B}{B}$ *(Modus Ponens)*
2. $\dfrac{A(v) \rightarrow B}{(\exists v)A(v) \rightarrow B}$, such that there are no free occurrences of v in B.
3. $\dfrac{B \rightarrow A(v)}{B \rightarrow (\forall v)A(v)}$, such that there are no free occurrences of v in B.

Figure 2: Inference Rules for the Predicate Calculus $Q\tau$

In [CCdS], Annotated Logic was investigated as a formal model that can be used to represent various types of incomplete and uncertain information. The fundamental idea is that it can be properly "instantiated' to represent specific formalisms and consequently adapted to simulate different automated reasoning systems in Artificial Intelligence. This "instantiation" is done through the specification of the lattice of truth values τ and of the function $\sim$.

As an example, a simplified version of necessity measures [DP87] is presented in [CdS96] to reason *about* uncertainty. The language is based on a partially ordered structure $\mathcal{N}$ of values that assign a degree of *certainty* or *quality* to a given proposition: the greater the degree, the greater is the certainty or credibility of the associated information. So, the *necessity* associated to a proposition p is the degree to which p can be considered *necessarily* or *certainly* true.

This language is the annotated language itself, where τ is instantiated as the structure $\mathcal{N}$ and the function $\sim$: $\mathcal{N} \to \mathcal{N}$ is an arbitrary weak negation function conveying the intuitive meaning of "not being certain" about a statement. It allows solving problems such as "*if the necessity of A is n_1 $(A : n_1)$, and the necessity of A being n_2 implies that the necessity of B is n_3 $(A : n_2 \to B : n_3)$, then can we conclude that B has necessity n_4, such that the value of n_4 is constrained by the values of n_1, n_2, n_3?*".

3 A Two-Sorted Reconstruction of $Q\tau$

In [CCdS97] it was suggested that, since Annotated Logic had an explicit (i.e. syntactic) representation of truth-values, it was useful to reason about truth valuations of arguments, thus it was a language to reason about arguments rather than to reason about objective facts.

This is akin to what is suggested in [Vis97], who argues for correspondence (i.e. relational) theories of truth as opposed to "disquotational" ones. Very briefly, this means that

$$\text{"}p\text{" is true} \leftrightarrow p$$

is not always necessarily the case. Hence, given p, we may still be interested about the relation between *"p"* and its truth-valuation. If we generalise the set of possible valuations for *"p"* to a finite lattice of truth values, then we can present Annotated Logic as a formal example to support this argument. Furthermore, if we consider – as proposed in the initial presentations of Annotated Logic [Sub87] – that this lattice of truth values characterise degrees of belief on logical statements, then we can introduce Annotated Logic as a correspondence theory of uncertainty.

A direct consequence of this is that Annotated Logic then becomes a logic of relations between logical statements and their truth-valuations (or degrees of belief). By replacing the word "relations" by "predicates", we feel tempted to reconstruct Annotated Logic as a predicate language whose terms are object-level relations and their corresponding truth values. The resulting language, however, would no longer be first-order.

Alternatively, we reconstruct this Logic as a two-sorted, first-order predicate language [Coh89, Coh87, EFT94, Gal86], in which one sort refers to object-level relations and the

other sort refers to truth values. This reconstruction is very straightforward and can be presented as follows.

The syntax of the language is given by:

1. A set of *individual variables*, defined as the union of the following sets:
   - A countable set of variable symbols, named *individual annotation variables*;
   - A countable set of variable symbols, named *individual object variables*;
2. Logical connectives $\neg, \wedge, \vee, \rightarrow$;
3. Quantifiers $\forall, \exists$;
4. Equality $=$;
5. Auxiliary symbols: parentheses, comma;
6. A set of *function* symbols, defined as the union of the following sets:
   - For each natural numbers $m, n < \omega$, a collection of functional annotation symbols of arity $m + n$. Functional annotation symbols of arity 0 are also called *individual annotations*;
   - For each natural numbers $m, n < \omega$, a collection of functional object symbols of arity $m + n$. Functional object symbols of arity 0 are also called *individual constants*;
7. For each natural numbers $m, n < \omega$, a collection of predicate symbols of arity $m + n$.

Terms are inductively defined as individual variables or applications of function symbols on terms. An *annotation term* is an individual annotation variable or the application of a functional annotation symbol, otherwise it is an *object term*. A term is *well-sorted* if it is an individual variable or a function symbol of arity $m + n$ applied to terms $t_1, ..., t_m, t_{m+1}, ..., t_{m+n}$ in which $t_1, ..., t_m$ are object terms and $t_{m+1}, ..., t_{m+n}$ are annotation terms.

Formulae are defined in the usual way. A formula is *well-sorted* if every predicate symbol of arity $m + n$ occurring in it is applied to terms $t_1, ..., t_m, t_{m+1}, ..., t_{m+n}$ and $t_1, ..., t_m$ are object terms and $t_{m+1}, ..., t_{m+n}$ are annotation terms.

The semantics of this language is based on two disjoint domains $\mathcal{A}$ and $\mathcal{O}$, respectively called annotation structure and object structure.

Definition 3.1 (Two-sorted Structure) *A two-sorted structure $\mathcal{A}, \mathcal{O}$ is formed by the following objects:*

1. *A non-empty set $|\mathcal{A}|$, named annotation domain.*
2. *A non-empty set $|\mathcal{O}|$, named object domain.*
3. *For each functional annotation symbol f_a, a function $f^{\mathcal{A}} : |\mathcal{O}|^m \times |\mathcal{A}|^n \rightarrow |\mathcal{A}|$.*
4. *For each functional object symbol f_o, a function $f^{\mathcal{O}} : |\mathcal{O}|^m \times |\mathcal{A}|^n \rightarrow |\mathcal{O}|$.*
5. *For each predicate symbol p, a function $p^{\mathcal{A},\mathcal{O}} : |\mathcal{O}|^m \times |\mathcal{A}|^n \rightarrow \{true, false\}$.*

Concepts such as *valuations* for variables and terms, *satisfiability*, *validity* and *semantic consequence* for well-sorted formulae are defined as usual, respecting the two sorts $\mathcal{A}$ and $\mathcal{O}$.

The axioms and inference rules for this logic are the classical ones – also respecting the sorts.

This concludes the presentation of our classical two-sorted first-order language. From this language, we now formulate a theory corresponding to the annotated logic $Q\tau$. We start by selecting the function and predicate symbols to be as follows:

1. The only functional annotation symbols occurring in the theory are:
   (a) individual annotations;
   (b) $\sim$, such that the arity of $\sim$ is $0+1$, i.e. it is a unary function from annotations to annotations; and
   (c) $\sqcup$, such that the arity of $\sqcup$ is $0+2$, i.e. it is a binary function from annotations to annotations;

   As we will see in the axiomatisation of this theory, $\sim$ will be employed to represent the weak negation and $\sqcup$ will be employed to represent the least upper bound of the lattice of annotations. Two selected and distinct individual annotations – denoted as $\top$ and $\bot$ – are assumed to belong to the set of functional annotation symbols of the theory;

2. All the predicates occurring in the theory have arity $m+1, m \geq 0$, except for the predicate $\leq$ of arity $0+2$. It is a binary predicate on annotations used to represent partial order between annotation terms as expected; all other predicates in the theory have arity.

The axioms for this theory are presented in figure 3.

Let A, B, C be arbitrary formulae, F, G be arbitrary formulae with at least one occurrence of one of $\wedge, \vee, \rightarrow$ or one of the quantifiers, p be a predicate symbol, $t_i, 1 \leq i \leq n$ be terms of appropriate sort, λ, μ, μ_i, $1 \leq i \leq m$ be individual annotation variables and u, v individual object variables.

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
3. $((A \rightarrow B) \rightarrow A) \rightarrow A$
4. $(A \wedge B) \rightarrow A$
5. $(A \wedge B) \rightarrow B$
6. $A \rightarrow (B \rightarrow (A \wedge B))$
7. $A \rightarrow (A \vee B)$

8. $B \to (A \vee B)$

9. $(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$

10. $(F \to G) \to ((F \to \neg G) \to \neg F)$

11. $F \to (\neg F \to A)$

12. $F \vee \neg F$

13. $A[t] \to (\exists v)A(v)$

14. $(\forall v)A(v) \to A[t]$

15. $p(t_1, ..., t_n, \bot)$

16. $\forall \mu, \lambda\ (p(t_1, ..., t_n, \mu) \wedge \lambda \leq \mu \to p(t_1, ..., t_n, \lambda))$

17. $\forall \mu\ ((\neg^k p(t_1, ..., t_n, \mu)) \leftrightarrow (\neg^{k-1} p(t_1, ..., t_n, \sim \mu))),\ k \geq 1$

18. $\forall \mu, \lambda\ ((p(t_1, ..., t_n, \mu) \wedge p(t_1, ..., t_n, \lambda)) \to p(t_1, ..., t_n, \mu \sqcup \lambda))$

19. $v = v$

20. $\mu = \mu$

21. $(v = u) \to (A(v) \leftrightarrow A_v'[u])$,
where $A_v'[u]$ is the formula that results from substituting u for some of the free occurrences of v in A such that u is replaceable for v in A

22. $(\mu = \lambda) \to (A(\mu) \leftrightarrow A_\mu'[\lambda])$,
where $A_\mu'[\lambda]$ is the formula that results from substituting λ for some of the free occurrences of μ in A such that λ is replaceable for μ in A

23. $\exists \mu_1, ..., \mu_m \forall \lambda\ (\lambda = \mu_1 \vee ... \vee \lambda = \mu_m)$

24. $\forall \mu_1, \mu_2 \exists \mu_3 \forall \lambda\ (\mu_3 \leq \mu_1 \wedge \mu_3 \leq \mu_2 \wedge ((\lambda \leq \mu_1 \wedge \lambda \leq \mu_2) \to \lambda \leq \mu_3))$

25. $\forall \mu_1, \mu_2, \lambda\ (\mu_1 \leq (\mu_1 \sqcup \mu_2) \wedge \mu_2 \leq (\mu_1 \sqcup \mu_2) \wedge ((\mu_1 \leq \lambda \wedge \mu_2 \leq \lambda) \to (\mu_1 \sqcup \mu_2) \leq \lambda))$

26. $\forall \mu\ (\mu \leq \top)$

27. $\forall \mu\ (\bot \leq \mu)$

28. $\forall \mu\ (\mu \leq \mu)$

29. $\forall \mu, \lambda\ ((\mu \leq \lambda \wedge \lambda \leq \mu) \to \mu = \lambda)$

30. $\forall \mu_1, \mu_2, \mu_3\ ((\mu_1 \leq \mu_2 \wedge \mu_2 \leq \mu_3) \to \mu_1 \leq \mu_3)$

Figure 3: Axiom System for the Two-sorted Theory that Implements $Q\tau$

Intuitively, an annotated atom $p(t_1, ..., t_m) : \mu$ in $Q\tau$ is denoted by $p(t_1, ..., t_m, \mu)$ in the present language, where $t_1, ..., t_m$ are object terms and μ is an annotated term. The translation of the axiomatisation of $Q\tau$ to an axiom system for this Two-sorted theory is straightfoward, as we can see from axioms 15 to 18 in both theories (figures 1 and 3).

It is interesting to observe the interpretation of negation in this theory, ruled by axiom 17. Following this axiom, we have that negation is interchanged with the application of the function $\sim$, thus it is better understood as a function in the theory than as a connective[2] – which is an immediate consequence of treating truth values relationally. This can be an explanation to why the proper treatment of negation is so distinct from that of other connectives in deductive systems – especially the non-classical ones.

Axioms 23 to 30 are introduced to ensure that the annotations will form a finite lattice and that $\leq$ and $\sqcup$ will convey the intended intuitive meaning for partial order and least upper bound of the lattice.

Implementing a clausal version of the first-order theory presented here as e.g. a PROLOG program is straightforward.

4 Discussion

We have proposed a reconstruction of the Annotated Logic $Q\tau$ as a two-sorted, first-order classical language. The interest in this reconstruction is three-fold:

1. the Logic $Q\tau$ thus presented clarifies its relation with a correspondence theory of degrees of belief, hence contributing to the philosophical debate on logical theories for uncertain reasoning;

2. there exist many efficient theorem provers for first-order classical theories, e.g. based on Resolution. This presentation of $Q\tau$ makes its implementation using those theorem provers straightforward;

3. dealing with negative information is a complex philosophical and practical issue in many logical systems, including most systems to represent uncertainty. The suggestion that negation should be considered a higher-order function rather than a connective, supported by this presentation of $Q\tau$, may shed some light on why it is so difficult to reason with negative information.

Future articles shall be devoted to more thorough discussions of each of these points.

Acknowledgements: The authors are partially supported by FAPESP, CAPES and CNPq. The second author is supported by CNPq grant 200074/97-0.

References

[Abe92] J. M. Abe. *Fundamentos de Logica Anotada (Foundations of Annotated Logic) – in portuguese.* PhD thesis, University of São Paulo, 1992.

[2] Although it will be a second-order function in this first-order theory.

[Abe93] J. M. Abe. On Annotated Model Theory. Coleção DOCUMENTOS - Lógica e Teoria da Ciência 11, Instituto de Estudos Avançados, Universidade de São Paulo, Junho 1993.

[BS87] H. A. Blair and V. S. Subrahmanian. Paraconsistent Logic Programming. *Theoretical Computer Science*, 68, 1987.

[BS88] H. A. Blair and V. S. Subrahmanian. Paraconsistent Foundations for Logic Programming. *The Journal of Non-Classical Logic*, 5(2), November 1988.

[Car96] D. V. Carbogim. *Programacao em Logica Anotada: Teoria e Aplicacoes (MSc dissertation)*. 1996.

[CAS91] N. C. A. Costa, J. M. Abe, and V. S. Subrahmanian. Remarks on Annotated Logic. *Zeitschr. f. Math. Logik und Grundlagen d. Math.*, 37:561–570, 1991.

[CCdS] D. V. Carbogim and F. S. Corrêa da Silva. Annotated Logic Applications for Imperfect Information. *Applied Intelligence (forthcoming)*.

[CCdS97] D. V. Carbogim and F. S. Correa da Silva. Facts, Arguments, Annotations and Reasoning. *New Generation Computing (submitted)*, 1997.

[CdS96] F. S. Correa da Silva. On Reasoning With and Reasoning About Uncertainty in Artificial Intelligence. In *European Summer Meeting of the Association of Symbolic Logic (abstract published in The Bulletin of Symbolic Logic, v. 3 (2), pp. 255-256)*, Spain, 1996.

[Coh87] A. G. Cohn. A More Expressive Formulation of Many-sorted Logic. *Journal of Automated Reasoning*, 3:113–200, 1987.

[Coh89] A. G. Cohn. Taxonomic Reasoning with Many-sorted Logics. *Artificial Intelligence Review*, 3:89–128, 1989.

[Cos63] N. C. A. Costa. *Sistemas Formais Inconsistentes*. PhD thesis, Universidade Federal do Parana (tese de catedra), 1963.

[Cos74] N. C. A. Costa. On the Theory of Inconsistent Formal Theories. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[CSV91] N. C. A. Costa, V. S. Subrahmanian, and C. Vago. The Paraconsistent Logics $P\tau$. *Zeitschr. f. Math. Logik und Grundlagen d. Math.*, 37:139–148, 1991.

[DP87] D. Dubois and H. Prade. Necessity Measures and the Resolution Principle. Technical Report 267, LSI - Universite Paul Sabatier, 1987.

[EFT94] H. D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag (2nd. ed), 1994.

[Gal86] J. H. Gallier. *Logic for Computer Science*. Harper & Row, Publishers, Inc., 1986.

[KL92] M. Kifer and E. L. Lozinskii. A Logic for Reasoning with Inconsistency. *Journal of Automated Reasoning*, 9:179–215, 1992.

[KS89] M. Kifer and V. S. Subrahmanian. On the Expressive Power of Annotated Logic Programs. In *NACLP'89 - Proceedings of the 1989 North American Conference on Logic Programming*, 1989.

[KS92] M. Kifer and V. S. Subrahmanian. Theory of Generalized Annotated Logic Programs and its Applications. *Journal of Logic Programming*, 12:335–367, 1992.

[NS90a] R. Ng and V. S. Subrahmanian. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases. Technical Report 2563, University of Maryland, Department of Computer Science, 1990.

[NS90b] R. Ng and V. S. Subrahmanian. Stable Semantics for Probabilistic Deductive Databases. Technical Report 2573, University of Maryland, Department of Computer Science, 1990.

[NS91] R. Ng and V. S. Subrahmanian. Relating Dempster-Shafer Theory to Stable Semantics. Technical Report 2647, University of Maryland, Department of Computer Science, 1991.

[NS92] R. Ng and V. S. Subrahmanian. Probabilistic Logic Programming. *Information and Computation (forthcoming)*, 1992.

[Sub87] V. S. Subrahmanian. On the Semantics of Quantitative Logic Programs. In *Proc. 4th IEEE Symposium on Logic Programming*, pages 173–182, San Francisco, September 1987.

[Thi95] K. Thirunarayan. On the Relationship between Annotated Logic Programs and Nonmonotonic Formalisms. *J. Expt. Theor. Artif. Intell.*, 7:391–406, 1995.

[Vis97] G. Vision. Why Correspondence Truth will not Go Away. *Notre Dame Journal of Formal Logic*, 38:104–131, 1997.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 1994 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail(mac@ime.usp.br).

Thomaz L. Seidman e Carlos Humes Jr.
SOME KANBAN-CONTROLLED MANUFACTURING SYSTEMS: A FIRST STABILITY ANALYSIS
RT-MAC-9501, janeiro de 1995, 19 pp.

C.Humes Jr. and A.F.P.C. Humes
STABILIZATION IN FMS BY QUASI- PERIODIC POLICIES
RT-MAC-9502, março de 1995, 31 pp.

Fabio Kon e Arnaldo Mandel
SODA: A LEASE-BASED CONSISTENT DISTRIBUTED FILE SYSTEM
RT-MAC-9503, março de 1995, 18 pp.

Junior Barrera, Nina Sumiko Tomita, Flávio Soares C. Silva, Routo Terada
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY PAC LEARNING
RT-MAC-9504, abril de 1995,16 pp.

Flávio S. Corrêa da Silva e Fabio Kon
CATEGORIAL GRAMMAR AND HARMONIC ANALYSIS
RT-MAC-9505, junho de 1995,17 pp.

Henrique Mongelli e Routo Terada
ALGORITMOS PARALELOS PARA SOLUÇÃO DE SISTEMAS LINEARES
RT-MAC-9506, junho de 1995, 158 pp.

Kunio Okuda
PARALELIZAÇÃO DE LAÇOS UNIFORMES POR REDUÇÃO DE DEPENDÊNCIA
RT-MAC-9507, julho de 1995, 27 pp.

Valdemar W. Setzer e Lowell Monke
COMPUTERS IN EDUCATION: WHY, WHEN, HOW
RT-MAC-9508, julho de 1995, 21 pp.

Flávio S. Corrêa da Silva
REASONING WITH LOCAL AND GLOBAL INCONSISTENCIES
RT-MAC-9509, julho de 1995, 16 pp.

Julio M. Stern
MODELOS MATEMÁTICOS PARA FORMAÇÃO DE PORTFÓLIOS
RT-MAC-9510, julho de 1995, 43 pp.

Fernando Iazzetta e Fabio Kon
A DETAILED DESCRIPTION OF MAXANNEALING
RT-MAC-9511, agosto de 1995, 22 pp.

Flávio Keidi Miyazawa e Yoshiko Wakabayashi
POLYNOMIAL APPROXIMATION ALGORITHMS FOR THE ORTHOGONAL Z-ORIENTED 3-D PACKING PROBLEM
RT-MAC-9512, agosto de 1995, pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON COLLECTIONS OF CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MORPHOLOGICAL MACHINES
RT-MAC-9513, agosto de 1995, 84 pp.

Marco Dimas Gubitoso e Jörg Cordsen
PERFORMANCE CONSIDERATIONS IN VOTE FOR PEACE
RT-MAC-9514, novembro de 1995, 18pp.

Carlos Eduardo Ferreira e Yoshiko Wakabayashi
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS COMBINATÓRIOS: TEORIA, ALGORITMOS E APLICAÇÕES
RT-MAC-9515, novembro de 1995, 45 pp.

Markus Endler and Anil D'Souza
SUPPORTING DISTRIBUTED APPLICATION MANAGEMENT IN SAMPA
RT-MAC-9516, novembro de 1995, 22 pp.

Junior Barrera, Routo Terada,
Flávio Corrêa da Silva and Nina Sumiko Tomita
*AUTOMATIC PROGRAMMING OF MMACH'S FOR OCR**
RT-MAC-9517, dezembro de 1995, 14 pp.

Junior Barrera, Guillermo Pablo Salas and Ronaldo Fumio Hashimoto
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMING OF MMACH'S
RT-MAC-9518, dezembro de 1995, 14 pp.

Daniela V. Carbogim and Flávio S. Corrêa da Silva
FACTS, ANNOTATIONS, ARGUMENTS AND REASONING
RT-MAC-9601, janeiro de 1996, 22 pp.

Kunio Okuda
REDUÇÃO DE DEPENDÊNCIA PARCIAL E REDUÇÃO DE DEPENDÊNCIA GENERALIZADA
RT-MAC-9602, fevereiro de 1996, 20 pp.

Junior Barrera, Edward R. Dougherty and Nina Sumiko Tomita
AUTOMATIC PROGRAMMING OF BINARY MORPHOLOGICAL MACHINES BY DESIGN OF STATISTICALLY OPTIMAL OPERATORS IN THE CONTEXT OF COMPUTATIONAL LEARNING THEORY.
RT-MAC-9603, abril de 1996, 48 pp.

Junior Barrera e Guillermo Pablo Salas
SET OPERATIONS ON CLOSED INTERVALS AND THEIR APPLICATIONS TO THE AUTOMATIC PROGRAMMINIG OF MMACH'S
RT-MAC-9604, abril de 1995, 66 pp.

Kunio Okuda
CYCLE SHRINKING BY DEPENDENCE REDUCTION
RT-MAC-9605, maio de 1996, 25 pp.

Julio Stern, Fabio Nakano e Marcelo Lauretto
REAL: REAL ATTRIBUTE LEARNING FOR STRATEGIC MARKET OPERATION
RT-MAC-9606, agosto de 1996, 16 pp.

Markus Endler
SISTEMAS OPERACIONAIS DISTRIBUÍDOS: CONCEITOS, EXEMPLOS E TENDÊNCIAS
RT-MAC-9607, agosto de 1996, 120 pp.

Hae Yong Kim
CONSTRUÇÃO RÁPIDA E AUTOMÁTICA DE OPERADORES MORFOLÓGICOS E EFICIENTES PELA APRENDIZAGEM COMPUTACIONAL
RT-MAC-9608, outubro de 1996, 19 pp.

Marcelo Finger
NOTES ON COMPLEX COMBINATORS AND STRUCTURALLY-FREE THEOREM PROVING
RT-MAC-9609, dezembro 1996, 28 pp.

Carlos Eduardo Ferreira, Flávio Keidi Miyazawa e Yoshiko Wakabayashi (eds)
ANAIS DA I OFICINA NACIONAL EM PROBLEMAS DE CORTE E EMPACOTAMENTO
RT-MAC-9610, dezembro de 1996, 65 pp.

Carlos Eduardo Ferreira, C. C. de Souza e Yoshiko Wakabayashi
REARRANGEMENT OF DNA FRAGMENTS: A BRANCH-AND-CUT ALGORITHM
RT-MAC-9701, janeiro de 1997, 24 pp.

Marcelo Finger
NOTES ON THE LOGICAL RECONSTRUCTION OF TEMPORAL DATABASES
RT-MAC-9702, março de 1997, 36 pp.

Flávio S. Corrêa da Silva, Wamberto W. Vasconcelos e David Robertson
COOPERATION BETWEEN KNOWLEDGE BASED SYSTEMS
RT-MAC-9703, abril de 1997, 18 pp.

Junior Barrera, Gerald Jean Francis Banon, Roberto de Alencar Lotufo, Roberto Hirata Junior
MMACH: A MATHEMATICAL MORPHOLOGY TOOLBOX FOR THE KHOROS SYSTEM
RT-MAC-9704, maio de 1997, 67 pp.

Julio Michael Stern e Cibele Dunder
PORTFÓLIOS EFICIENTES INCLUINDO OPÇÕES
RT-MAC-9705, maio de 1997, 29 pp.

Junior Barrera e Ronaldo Fumio Hashimoto
COMPACT REPRESENTATION OF W-OPERATORS
RT-MAC-9706, julho de 1997, 13 pp.

Dilma M. Silva e Markus Endler
CONFIGURAÇÃO DINÂMICA DE SISTEMAS
RT-MAC-9707, agosto de 1997, 35 pp

Kenji Koyama e Routo Terada
AN AUGMENTED FAMILY OF CRYPTOGRAPHIC PARITY CIRCUITS
RT-MAC-9708, setembro de 1997, 15 pp

Routo Terada e Jorge Nakahara Jr.
LINEAR AND DIFFERENTIAL CRYPTANALYSIS OF FEAL-N WITH SWAPPING
RT-MAC-9709, setembro de 1997, 16 pp

Flávio S. Corrêa da Silva e Yara M. Michelacci
MAKING OF AN INTELLIGENT TUTORING SYSTEM (OR METHODOLOGICAL ISSUES OF ARTIFICIAL INTELLIGENCE RESEARCH BY EXAMPLE)
RT-MAC-9710, outubro de 1997, 16 pp.

Marcelo Finger
COMPUTING LIST COMBINATOR SOLUTIONS FOR STRUCTURAL EQUATIONS
RT-MAC-9711, outubro de 1997, 22 pp.

Maria Angela Gurgel and E.M. Rodrigues
THE F-FACTOR PROBLEM
RT-MAC-9712, dezembro de 1997, 22 pp.

Perry R. James, Markus Endler, Marie-Claude Gaudel
DEVELOPMENT OF AN ATOMIC-BROADCAST PROTOCOL USING LOTOS
RT-MAC-9713, dezembro de 1997, 27.pp.

Carlos Eduardo Ferreira and Marko Loparic
A BRANCH-AND-CUT ALGORITHM FOR A VEHICLE ROUTING PROBLEM WITH CAPACITY AND TIME CONSTRAINTS
RT-MAC-9714, dezembro de 1997, 20 pp.

Nami Kobayashi
A HIERARCHY FOR THE RECOGNIZABLE M-SUBSETS
RT-MAC-9715, dezembro de 1997, 47 pp.

Flávio Soares Corrêa da Silva e Daniela Vasconcelos Carbogim
A TWO-SORTED INTERPRETATION FOR ANNOTATED LOGIC
RT-MAC-9801, fevereiro de 1998, 17 pp.