# ANNOTATED LOGIC REASONING BASED ON XML

Fuxi Zhu, Duanzhi Chen, Jia Rao
*School of Computer, Wuhan University, Wuhan, China 430072*

Keywords: Paraconsistent Logic, Annotate Logic, XML, Automatic Reasoning.

Abstract: Paraconsistent logic is able to handle the inconsistent knowledge reasonably. In this paper we propose using XML as a tool to implement the presentation and reasoning of Annotation Logic--one of the paraconsitent logic systems and investigate the problems about the automatic inference rules and inference strategies under XML representation. Demonstrations are presented to show that XML can represent the Annotation Logic conveniently and XML and its auxiliary tools can implement the inference mechanism efficiently.

## 1 INTRODUCTION

Inconsistency appears very often among the various knowledge discovery systems in practice. The knowledge which comes from different information source may contradict each other. If the methods of extraction of knowledge or the expression of knowledge are inappropriate, or the knowledge is beyond the validation scope, it will cause the knowledge distortion or incompatibility. The most simple situation is, if we obtain assertion A from knowledge source s1, obtains assertion B from information source s2, and obtain assertion ~A∨ ~B from information source s3, that means A and B is impossibly true at the same time. If these three assertions are put together, then an inconsistence occurs immediately.

It is not difficult to see that, as long as we extract knowledge from the different sources, the incompatibility processing is inevitable. But the classical logic is nearly helpless to this kind of incompatibility. For in classical logic, so long as the contradiction exists in knowledge base, any conclusion will be induced. Thus, a huge knowledge base could be collapsed because of its containing a small inconsistent subset. Therefore, the research in this area has the great significance.

At present, the paraconsistent logic research is being valued increasingly by logician and computer scientists.

Brazilian logician da Costa proposed a set of formal paraconsistent calculating system C(Da Costa., 1992). The system C achieves the paraconsistence by weakening the classical logic deduction ability.

Following the idea of da costa, Kifer.M and Subrahmanan.V.S put forward the Annotated Logic(Kifer, M. 1992,).

M. Kifer and E. L. Lozinskii then further modify the Annotated Logic system into APC (Annotated Logic Calculus)( Kifer, M., Lozinskii, E.L. 1992).

In this article we take XML as a basic tool, and implement the representation and the reasoning of Annotated Logic

## 2 REPRESENT THE ANNOTATED LOGIC WITH XML

The Annotated Logic is one of mature paraconsistent logic formal systems. This logic introduces a set of annotated truth values(context) τ in the language L. If φ is a formula of language L and λ is a annotated truth value, then φ:λ is a annotated formula. In the Annotated Logic, it further t limits τ as a standard complete lattice. For example, set τ=Four= {t, F, ⊤ , ⊥ }. In this lattice, t and f stand for "true" and "false"; ⊤ stands for "contradiction"; ⊥ stands for "unknown". O, classical two-value logic is the specific case of the Annotated Logic.

**Definition 1** If A is an atom formula and u ∈ τ, then:

( 1 ) $A{:}u$ is an annotated atom.

( 2 ) ~ … ~ ($A{:}u$) is called the k order mega-literal, k means the symbol ~ appear k times.

Zhu F., Chen D. and Rao J. (2006).
ANNOTATED LOGIC REASONING BASED ON XML.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 229-233
DOI: 10.5220/0001241702290233

**Definition 2** Suppose $A_0, A_1 \ldots A_n$ are the headers of a literal, $\lambda_0, \lambda_1, \ldots, \lambda_n$ are annotated values. A mega-clause is a formula in the following form :

$(\forall x_1) \ldots (\forall x_n) A_0{:}\lambda_0 \vee A_1{:}\lambda_1 \vee \ldots \vee A_n{:}\lambda_n$

Where, $x_1, x_2 \ldots x_n$ are variables appeared in $A_i$; $A_i{:}\lambda_i$ is a mega-literal whose order is smaller than or equal to 1.

Because the XML language has strong capability of semantics expression, we use XML to implement the representation and reasoning of Annotated Logic. For annotated literal φ:λ, its full expression is:

sign predicate_name(argument_list):value

With the XML language, it can be express as

```
<annotated_predicate>
  <predicate_sign> sign </predicate_sign>
  <predicate_name> name </predicate_name>
  <argument_list>
      <argument>arg1</argument>
      <argument>arg2</argument>
        ……
  </argument_list>
  <annotated_value> val </annotated_value>
</annotated_predicate>
```

In the section above, *val* belongs to Four = {t, f, ⊤ , ⊥ }.

The mega-clause that defined in Definition 2 is easily expressed with XML as:

```
<annotated_clause>
  <annotated_predicate>
    …
  </annotated_predicate>
  <annotated_predicate>
    …
  </annotated_predicate>
  …
</annotated_clause>
```

A paraconsistent knowledge base under Annotated Logic composes of many mega-clauses. Thus it can be defined with XML as:

```
<knowledge_base>
  <annotated_clause>
    …
  </annotated_predicate>
  <annotated_clause>
    …
  </annotated_clause>
  …
</knowledge_base>
```

So, a XML document can express a paraconsistent knowledge base. To standardize the expression of the parameters of a mega-literal, mega-clause and the knowledge base, we use XML schema to define XML document that represent a knowledge base. Thus we can implement the various inference mechanisms on this knowledge base.

# 3 THE AUTOMATIC REASONING OF ANNOTATED LOGIC WITH THE XML EXPRESSION

The conception of unifying also exists in the process of refutation between the formulae of Annotated Logic. It is similar to the unifying of the formulae of the classical predicate logic. Under the expression of XML, the selection process of parent clauses can be implemented as the process of search and matching in the XML document. Thus, we will discuss the inference mechanism of Annotated Logic and how to implement it.

**Definition 3** set $C_1$ and $C_2$ are mega-clauses as following:

$C_1 \equiv (L_1 \vee \ldots \vee L_i \vee \ldots \vee L_n)$

$C_2 \equiv (L_1' \vee \ldots \vee L_j' \vee \ldots \vee L_m')$

Where $L_i = (A{:}u)$, $L_j' = \sim (A'{:}\rho)$, $A$ and $A'$ are able to unify via a mgu $\theta$, $u$, $p \in \tau$ and $p \leqq u$, thus the resolvent of refutation $C_1$ to $C_2$ via mega-literal $L_i$ and $L'_j$ can be defined as:

$(L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \ldots L_n \vee L_1' \vee \ldots \vee L_{j-1}' \vee L_{j+1}' \vee \ldots \vee L'_m)\theta$

Where $\theta$ is called unifier of mega-refutation step, $L_i$ and $L'_j$ are mega-literals which are deleted at mega-refutation step.

In Definition 3, it requires that the relation between the annotated value $u$ of 0 order literal and the annotated value $p$ of 1 order literal satisfies $p \leqq u$.

For a set of mega-clauses $S$, a series of mega-refutations between the mega-clauses of $S$ and the results of a mega-refutation are called a mega-inference.

To realize mega-inference, Java package JDom is used to parse the XML document which contains the expression of the set of mega-clauses and to produce the Document object, and then the *getChildren*("mega-clause") method of the root Document object is called to extract all the mega-clauses from the knowledge base. These mega-clauses constitute the initiative mega-clause set. Its structure is a *List*. The target clauses which we want to deduce is also in this List. The inference process is just using the *get*(int *i*) method of the *List* to select a clause from the set of mega-clause.

It must be noted that the mega-refutation do not possess the completeness. The following case is an example to illustrate the incompleteness of mega-refutation.

**Example 1** set $S$={ p:t, p:f, ~ (p:⊤ )}, $S$ has no model. But its mega-inference of null clause □ from $S$ does not exist. For ⊤ ≧ f and ⊤ ≧ t, ~(p:⊤ ) can not refute with neither p:t nor p:f.

This example told us that, to ensure the completeness of mega-inference, we have to continue to prefect the inference mechanism.

**Definition 4** Suppose $u,u_1,u_2\in \tau$, if $u=\cup \{u_1,u_2\}$, i.e., $u$ is the least upper bound of $u_1$ and $u_2$, the pair ($u_1$, $u_2$) is called disassembly of $u$. ($u_1$, $u_2$) is called a strict disassembly of $u$ if ($u_1$, $u_2$) is a disassembly of $u$, and $u$, $u_1$, $u_2$ satisfy $u_1\neq u\neq u_2$.

In complete lattice Four, the strict disassembly of truth ⊤ are pair (t, f) and pair (f, t). Whereas t, f and ⊥ have no disassembly.

It is not difficult to proof the following predication.

**Proposition 1** suppose $A$ is a atom, $C$ and $D$ are mega-clauses and $u \in \tau$, ($u_1$, $u_2$) is a disassembly of $u$, then:

(1) $\models(A{:}u \Leftrightarrow (A{:}u_1 \,\&\, A{:}u_2))$ and

(2) $\models(C\vee \sim(A{:}u) \vee D) \Leftrightarrow (C\vee \sim(A{:}u_1)\vee \sim (A{:}u_2)\vee D)$

Considering Proposition 1 and the fact that only ⊤ can be disassembled in lattice Four, we further extend the Definition 3 and put forward the following definition.

**Definition 5** Suppose $C_1$ and $C_2$ are the mega-clauses in Definition 3, $L_i=(A{:}u)$, $L_j'=\sim(A'{:}p)$, $u\in$ {t, f}, $p$=⊤ , and $A$, $A'$ can unify via mgu $\theta$, then the binary mega-refutation $C_1$ to $C_2$ over mega-literal $L_i$ and $L'_j$ can be defined as:

$(L_1\vee \ldots\vee L_{i-1}\vee L_{i+1}\ldots L_n\vee L_1'\vee \ldots\vee L_{j-1}'\vee \sim(A{:}\neg u)\vee L_{j+1}'\vee \ldots\vee L'_m)\theta$

The purport of Definition 5 is to reduce some unnecessary steps of disassembly. Because of $L_j'=\sim(A'{:}\top) \Leftrightarrow (\sim(A'{:}f)\vee \sim(A'{:}t))$, it can refute with $(A{:}t)$ or $(A{:}f)$. The resolvent is $\sim(A{:}f)\theta$ or $\sim(A{:}t)\theta$. Hence, when two mega-clauses, one contains literal $L_i=(A{:}t)$ or $L_i=(A{:}f)$ and the other contains literal $L_j'=\sim(A'{:}\top)$, it can refute directly and no need to disassemble $\sim(A'{:}\top)$ and then to do refute.

**Definition 6** set $D\equiv(D_1\vee A_i{:}u_i\vee D_2\vee A_j{:}u_j\vee D_3)$, $D_1$, $D_2$, $D_3$ are mega-clauses. If $A_i$ can unify with $A_j$ via mgu $\theta$ and $u_i\geqq u_j$, $D'\equiv(D_1\vee A_i{:}u_i\vee D_2\vee D_3)\theta$ is called the factor of $D$.

Similarly, set $E\equiv D_1\vee \sim(A_i{:}u_i)\vee D_2\vee \sim(A_j{:}u_j)\vee D_3$ is a mega-clause, if $A_i$ can unify with $A_j$ via mgu $\theta$ and $u_i\geqq u_j$, $D''\equiv(D_1\vee D_2\vee \sim(A_j{:}u_j)\vee D_3)\theta$ is also called the factor of $E$.

As we saw above, there are two cases of definition of factor. One case is positive literal being merged and the other case is negative literal being merged. In the clause corresponding to the factor $D'$, $A_i$, $A_j$ are positive literal and $u_i\geqq u_j$. It indeed deletes the literal containing $u_j$, and $A_i{:}u_i$ is reserved in $D'$. In the clause corresponding to the factor $D''$, $\sim A_i{:}u_i$ and $\sim A_j{:}u_j$ are mega-literal, $u_i\geqq u_j$, the literal containing $u_i$ is deleted, and $\sim A_j{:}u_j$ is reserved in $D''$.

According to the Definition 6, we could leave out unnecessary mega-literals. For example, in $\sim(A_i{:}t)\vee \sim(A_j{:}\top)$, we can leave out $\sim(A_i{:}t)$.

**Proposition 2** If $C$ and $D$ are mega-clauses and $D$ is a factor of $C$, $C\models D$. $C\models D$ represents that $D$ is the logic conclusion of $C$, i.e., any model of $C$ is also the model of $D$.

**Definition7** (subsumption ) suppose $C_1$ and $C_2$ are mega-clauses in the following form :

$C_1\equiv A_1{:}u_1\vee \ldots\vee A_n{:}u_n\vee \sim(B_1{:}\rho_1)\vee \ldots\vee (B_m{:}\rho_m)$

$C_2\equiv G_1{:}\psi_1\vee \ldots\vee G_r{:}\psi_r\vee \sim(H1{:}\varphi_1)\vee \ldots\vee \sim(H_s{:}\varphi_s)$

$C_1$ subsumes $C_2$ if and only if exist a substitution $\theta$, so as to:

(1) For all $i$, $1\leq i\leq n$, exists $j$, $1\leq j\leq r$, makes $G_j=A_i\theta$ and $u_i\geqq \psi_j$.

(2) For all $l$, $1\leq l\leq m$, exists $w$, $1\leq w\leq s$, makes $H_w=B_l\theta$ and $\varphi_w\geqq \rho_l$.

The requirement of the Definition of subsumption is very strict. Mega-clause $C_1$ subsumeing $C_2$ requires that for all mega-literal $A_i{:}u_i$ of $C_1$, we must find a mega-literal $G_j{:}\psi_j$ in $C_2$ and a substitution $\theta$ make $G_j=A_i\theta$ and the annotated value of $G_j$ satisfy $u_i\geqq \psi_j$. Furthermore, for the mega-literal of $C_2$, the definition has the similar requirement. We need to find a mega-literal $H_w{:}\varphi_w$

in $C_2$ and the substitution $\theta$ make $H_w=B_l\theta$ and the annotated value of $H_w$ satisfy $\varphi_w \leqq \rho_l$.

**Proposition 3** If $C_1$ and $C_2$ are mega-clauses and $C_1$ subsumes $C_2$, then $C_1 \models C_2$.

The advantage the definition 7 and proposition 3 bringing to us is that we can delete the mega-clause in the process of mega-refutation. This action greatly reduces the size of *List* which contains the XML expression of the set of mega-clauses. If one more mega-clause is added the *List*, it may nearly double the size of *List* in the mega-refutation process.

**Definition 8** set $c_1$ and $c_2$ are mega-clauses, then mega-clause $C$ is called a paraconsistent refutation(p-refutation for short) if and only if $C$ is a binary mega-refutation of $C'_1$ to $C'_2$, where $C'_i$(i=1,2) are：

1) $C_i$ or
2) the factor of $C_i$ or
3) a mega-clause in $C''_i$, $C''_i$ is disassembly of $C_i$.

It is not difficult to make out that the Definition 8 comprises the meaning of Definition 5, 6. In the process of refutation, the factor is easy to get when extracting the clause from the *List*. If a disassembly is needed in step (3), the refutation can be finished according to Definition 5.

**Definition 9** An linear inference sequence $C_1$, $C_2$… $C_n$ from the mega-clause set $S$, satisfy:

1) $C_1 \in S$ and
2) for $i > 0$, $C_{\{i+1\}}$ is one of p-refutation $C_i$ to $C$.

Inference sequence $C_1$, $C_2$… $C_n$, is a para-inference of $S$ if and only if $C_n=\square$, where $\square$ is null clause or any clause in the following form:

$$(\sim(A_1:\perp) \vee \ldots \vee \sim(A_i:\perp))$$

The proof of completeness of para-inference in Definition 9 is similar to the proof of lifting-lemmas of first order predicate logic. This implement strategy of para-inference is also similar to the support set strategy of first order predicate logic. In the implement process, the selection of clauses from the clause set is to call the *get*(int *i*) method of *List*. The implementation of the unifying algorithm of two mega-literals is to call *get* method of XML Element and compare variable of the two literals one by one. In this process the Xpath technology is used to get all the variables and XLST technology is used to realize the substitution of variables. Xpath is also used to access the annotated value. Xpointer technology is used for the matching of parameters and locating the level of the sub-elements.

# 4 AN EXAMPLE OF REASONING

Take the medicine expert system as an example of paraconsistent reasoning, we construct a medicine expert system by consulting doctors. We suppose that doctor A provides us the following three rules ($S_i$ represents a symptom, $D_i$ represents a disease):

$C_1$: $D_1(x)$:t$\vee$ $D_2(x)$:t$\vee$ $\sim(S_1(x)$:t$)\vee$ $\sim(S_2(x)$:f$)$

$C_2$: $D_1(x)$:f$\vee$ $\sim(D_2(x)$:t$)$

$C_3$: $D_2(x)$:f$\vee$ $\sim(D_1(x)$:t$)$

Intuitively, this doctor tells us if one person is inspected have symptom $S_1$, and no symptom $S_2$, and then this patient suffers disease $D_1$ or disease $D_2$. Moreover, it tells us that no body suffers disease $D_1$ or disease $D_2$ simultaneity.

The information which Doctor B provides expressed with mega-clauses is as following:

$C_4$: $D_1(x)$:t$\vee$ $\sim(S_2(x)$:f$)\vee$ $\sim(S_3(x)$:t$)$

$C_5$: $D_2(x)$:f$\vee$ $\sim(S_3(x)$:f$)$

Now, a pathology doctor examines the patients Tom and Tim and describes the symptoms as following:

$C_6$: $S_1$(Tom):t
$C_7$: $S_1$(Tim):t
$C_8$: $S_2$(Tom):f
$C_9$: $S_2$(Tim ):f
$C_{10}$: $S_3$(Tom):t

The pathology doctor's above report tells us that Tom has the symptom $S_1$, $S_3$, but does not has the symptom $S_2$; Tom has the symptom $S_1$ but does not has the symptom $S_2$.

Base on these examination results, it is easy for us to prove that $D_1$(Tom):t is a logic conclusion of the set of mega-clause $\{C_1, \ldots, C_{10}\}$. The p-refutation of $\{C_1,\ldots, C_{10}\} \cup \{\sim D_1(\text{Tom}):t\}$ is shown as following:

$E_1$: $\sim(D_1$(Tom ):t) (initial inquiry)

$E_2$: $\sim(S_2$(Tom ):f$)\vee$ $\sim S_3$(Tom ):t (resolvent of $E_1$ and $C_1$)

$E_3$: $\sim(S_3$(Tom):t) (resolvent of $E_2$ and $C_8$)

$E_4$: (resolvent of $E_3$ and $C_{10}$)

Similarly, $D_2$(Tom):f is also a logic conclusion of $\{C_1, \ldots, C_{10}\}$. The p-refutation of $\{C_1, \ldots, C_{10}\}\cup \{D_2(\text{Tom}):f\}$ is shown as following:

$E_1$: $\sim(D_2$(Tom ):f) (initial inquiry)

$F_2$: $\sim(D_1$(Tom ):t) ($F_1$ and $C_3$)

The rest part of the p-refutation is the same as the steps from $E_1$ to $E_4$

Until now, all the knowledge of these inquiries involved is consistent. Now suppose that a third doctor has already provided the following information to us:

$C_{11}$:$D_2(x)$:tv ~ ($S_3(x)$:t)

Put the rules and "datum" that these three doctors and pathology doctor provides together, when inconsistence is needed to process. Because using the rules Doctor 3 provides and data pathology doctor provides may infer that Tom suffers disease $D_2$. Similarly, from the data Doctor 2 provides and rules pathology doctor provides we can infer that Tom suffers disease $D_1$. Thus, Tom suffers both disease D1 and disease D2. Therefore, when the rule $C_{11}$ in knowledge base is considered, it can infer out $D_1$(Tom):T. The steps of mega-inference of ~ ($D_1$(Tom ):T) are as following:

$G_1$: ~ ($D_1$(Tom):T) (initial inquiry)

$G_2$: ~ ($S_2$(Tom):f)∨ ~ $S_3$(Tom):tv ~ ($D_1$(Tom ):f) (Decomposes of $C_4$ and $G_1$)

$G_3$: ~ ($S_3$(Tom):t)∨ ~ ($D_1$(Tom):f) (resolvent of $G_2$ and $C_8$)

$G_4$: ~ ($D_1$(Tom):f) (resolvent of $G_3$ and $C_{10}$)

$G_5$: ~ ($D_2$(Tom ):t) (resolvent of $G_4$ and $C_2$)

$G_6$: ~ ($S_3$(Tom ):t) (resolvent of $G_5$ and $C_{11}$)

$G_7$: (resolvent of $G_6$ and $C_{10}$)

# 5 CONCLUSIONS AND FURTHER WORKS

In this paper, we investigate how to use XML to realize automatic reasoning of paraconsistent logic. As we know, it is easy to produce the inconsistence from knowledge discovery in WEB environment. Therefore it is an appropriate way to express and deal with coordinated logic. At present, XML is more and more popular in the WWW community, so using XML as a tool of Web knowledge discovery and the expression of incompatible knowledge is a very natural way.

In this paper, the prefect match of XML and Java is subtly used to realize the inference mechanism of Annotated Logic. In our implementation, the logic truth is limited on Lattice Four. Under the complete Lattice Four, the frame of Annotated logic was modified and inference rule is reconstructed. This modification does not lose the completeness. In the implementation of the reasoning system, Java Jdom package is used to parse and process the XML document, in which inconsistent knowledge is represented. Seeing from the process of knowledge acquirement and knowledge inference, the knowledge representation based on XML and with the help of Java Jdom has a sound readability and good efficiency of reasoning. It is a big step to made paraconsistent reasoning from theory towards practical application.

Our further research may concentrate on whether the automatic reasoning theory that under the classical logic come into existence under the Annotated Logic. For example, besides most basic binary mega-refutation, whether the hyper-refutation, negative hyper-refutation, unit-refutation and equality refutation inference mechanism or the strategy are applicable in Annotated Logic. If true, whether XML is a suitable tool to implement these inference mechanisms and strategies is also a research topic. Furthermore, the soundness and completeness of these mechanisms and strategies should be discussed and investigated too.

# REFERENCE

Da Costa, 1992. Automated Theorem Proving in Paraconsistent Logic Theory and Implementation. In *10th Int. Conf. on Automated Deduction. pp: 72-86.*

Kifer, M., Subrahmanian, V. S. 1992. Theory of Generalized Annotated Logic Programming and its Applications. *In Journal of Logic Programming, 12(4):335-368.*

Kifer, M., Lozinskii, E.L., 1992. A Logic for Reasoning with Inconsistency. *IN Journal of Automated Reasoning, 9(2): 179-215.*

Gui, Q.Q., Chen, Z.L., Zhu, F.X., 2002. *Paraconsisten Logic and Artificial Intelligence*, The press of Wuhan University.

Zhu, F. X., Liu, L. P., Fu, J.M. 1998. Automatic Reasoning in Paraconsisten Logic, In *J. of Wuhan University(Science Edition) 581 ~594.*