Foundations of Data and Knowledge-based Systems

ATMS – Assumption-based Truth Maintenance Systems

Franz Wotawa

Technische Universität Graz
IICM – Software Technology
Email: wotawa@ist.tugraz.at
Introduction

- Example
- Basic Definitions
- Algorithm
- Properties
- Extensions
- Bibliography
Example (I)

Propositional Theory $Th$

\[
\begin{align*}
    a, & \quad c, \\
    a \to b, & \quad c \to d, \\
    a \land c \to e, & \quad b \land d \to \bot.
\end{align*}
\]

$\bot, \to, \land$ designate falsity, implication, conjunction

**Theory** $Th$ is inconsistent!

\[
\begin{align*}
    a, a \to b & \models b \\
    c, c \to d & \models d \\
    a, c, a \land c \to e & \models e \\
    b, d, b \land d \to \bot & \models \bot
\end{align*}
\]

From $\bot$ follows everything!
Example (II)

Aim: Eliminate inconsistency!

Default Logic (Only normal defaults):

\[
\begin{align*}
\text{true} & \quad A, \\
A & \rightarrow b, \\
A \land C & \rightarrow e, \\
C & \rightarrow d, \\
b \land d & \rightarrow \bot.
\end{align*}
\]

Compute Extensions

(= Consistent subsets of a theory)

\{A, b\} and \{C, d\}
Example (III)

Using the ATMS

ATMS Node: \(\langle p, \{CSD_1, \ldots, CSD_n\}\rangle\)

\(CSD_i\ldots\) Consistent set of defaults (ATMS assumptions)

1. \(A\) \hspace{1cm} \(\langle A, \{\{A\}\}\rangle\)
2. \(C\) \hspace{1cm} \(\langle C, \{\{C\}\}\rangle\)
3. \(A \rightarrow b\) \hspace{1cm} \(\langle b, \{\{A\}\}\rangle\)
4. \(C \rightarrow d\) \hspace{1cm} \(\langle d, \{\{C\}\}\rangle\)
5. \(A \land C \rightarrow e\) \hspace{1cm} \(\langle e, \{\{A, C\}\}\rangle\)
6. \(b \land d \rightarrow \bot\) \hspace{1cm} \(\langle \bot, \{\{A, C\}\}\rangle\)
\hspace{1cm} \(\langle e, \{\}\rangle\)

\(e\) is no longer supported! \(b\) is supported by \(A\) and \(d\) is supported by \(C\).
Algorithm

1. Let $Th$ be a set of facts and rules.

2. If $Th$ is consistent exit the algorithm.

3. Otherwise, select a fact or rule $r$ from $Th$.

4. Remove $r$ from $Th$, i.e., $Th = Th \setminus \{r\}$, and goto step 2.

- Makes no differences between facts and rules
- Makes no differences between different facts.
- Not appropriate in some (important) cases.
Problem-solver Architecture

Problem Solver domain knowledge, inference procedures, sends inferences to ATMS.

ATMS determine what data are believed and disbelieved, use assumptions and justifications

Example:

Multiplier \( m_1 \) with behavior \( \neg ab(m_1) \to out(m_1) = in_1(m_1) \cdot in_2(m_1) \)

Problem solver knows that \( in_1(m_1) = 3, \) \( in_2(m_1) = 3, \) and the behavior. Under the assumption that \( \neg ab(m_1) \) the problem solver can conclude \( out(m_1) = 6. \)

Problem solver sends justification to ATMS:

\[
\Gamma(in_1(m_1) = 3) \land \Gamma(in_2(m_1) = 2) \land \\
\Gamma(\neg ab(m_1)) \to \Gamma(out(m_1) = 6)
\]

The data \( \Gamma(\neg ab(m_1)) \) is an assumption.
Behavior and structure  \( mult(C) \land \neg ab(C) \rightarrow out(C) = in_1(C) \cdot in_2(C), plus(C) \land \neg ab(C) \rightarrow out(C) = in_1(C) + in_2(C), mult(M1), mult(M2), mult(M3), plus(A1), plus(A2), in_1(M1) = a, in_2(M1) = c, \ldots \)

Assumptions  \( \neg ab(M1), \neg ab(M2), \neg ab(M3), \neg ab(A1), \neg ab(A2) \) denoted by \( NAB(M1), NAB(M2), NAB(M3), NAB(A1), NAB(A2) \).

Justifications  \( NAB(M1), in_1(M1) = 2, in_2(M1) = 3 \rightarrow out(M1) = 6, NAB(M1), out(M1) = 6, in_1(M1) = 2 \rightarrow in_2(M1) = 3, \ldots \)
Justifications send to the ATMS (only partially for forward propagation)
Definitions (I)

**Node**  An ATMS node corresponds to a problem-solver datum.

**Assumption**  A special node.

**Justification**  Describes how nodes are derived from other nodes.

\[ X_1, \ldots, X_n \Rightarrow X_{n+1} \]

where \( X_i \) are nodes and \( X_1, \ldots, X_n \) is the antecedence and \( X_{n+1} \) the consequent.

Justifications are Horn Clauses!

**Environment**  Is a set of assumptions.

**Context**  Is formed by a consistent environment and all nodes derived from it.

**Characterizing environment**  Minimal consistent environment from which a context can be derived.
Definitions (II)

- A node $n$ holds in an environment $E$ iff $n$ can be derived from $E$ and the current theory $Th$, i.e., $E \cup T \models n$.

- An environment $E$ is inconsistent if the false node ($\bot$) can be derived, i.e., $E \cup Th \models \bot$.

- Every node $n$ has assigned labels. A label (for $n$) is a set of consistent environments from which $n$ can be derived.

- **Task of the ATMS**: Compute node labels.
Definitions (III) - Label Properties

**Consistent** A label $L$ for node $n$ is consistent if all of its environments are consistent.

**Sound** A label $L$ for node $n$ is sound iff $n$ is derivable from every environment $E$ from $L$.

$$E \cup Th \models n$$

**Complete** A label $L$ for node $n$ is complete iff every consistent environment $E \not\in L$ for which $E \cup Th \models n$ is a superset of some $E' \in L$, i.e., $E' \subset E$.

**Minimal** A label $L$ for node $n$ is minimal iff for every element $E$ of $L$ there exists no subset $E' \subset E$ from which $n$ can be derived $E' \cup Th \models n$. 

12
Consequences

- **Task of the ATMS**: Compute minimal, consistent, sound, and complete labels for every node.

- A node $n$ is derivable from an environment $E$ if $E$ is element of the label or $E$ is a superset of any element of the label.

- A node has an empty label iff it is not derivable from a consistent set of assumptions.

- Contexts are determined by node labels.

- ATMS can handle multiple contexts at the same time.
Coffee Machine Example

Model

Request → request
Water → water
Beans → beans
request ∧ water ∧ beans → coffee
coffee → request
coffee → water
coffee → beans
Coffee Machine (II)

Model (cont.)

\[
\begin{align*}
\text{no\_coffee} \land \text{request} \land \text{water} & \rightarrow \text{no\_beans} \\
\text{no\_coffee} \land \text{request} \land \text{beans} & \rightarrow \text{no\_water} \\
\text{no\_coffee} \land \text{water} \land \text{beans} & \rightarrow \text{no\_request} \\
\text{beans} \land \text{no\_beans} & \rightarrow \perp \\
\text{request} \land \text{no\_request} & \rightarrow \perp \\
\text{water} \land \text{no\_water} & \rightarrow \perp \\
\text{coffee} \land \text{no\_coffee} & \rightarrow \perp \\
\end{align*}
\]

Observations

\[
\text{no\_coffee}
\]
Coffee Machine (III)

water  \{\{Water\}\}
beans  \{\{Beans\}\}
request \{\{Request\}\}
coffee \{}
no_coffee \{\{\}\}\}
nobean s  \{\{Water, Request\}\}
no_water \{\{Beans, Request\}\}
no_request \{\{Water, Beans\}\}
⊥        \{\{Water, Beans, Request\}\}
Coffee Machine (IV)

Add the fact `water` to the ATMS

```
| water       | {{}} |
| beans       | {{Beans}} |
| request     | {{Request}} |
| coffee      | {} |
| no_coffee   | {{}} |
| no_beans    | {{Request}} |
| no_water    | {} |
| no_request  | {{Beans}} |
| ⊥           | {{Beans, Request}} |
```

Only missing Beans or Request remains as source of the misbehavior, i.e., no_coffee.

What means no_beans `{{Request}}`? Under the assumption that Request is true no_beans must be valid.
Robotics

- ATMS for representing the state of the world.

- Different kind of ‘Facts’: (1) Real facts, (2) Currently valid assumptions

  The sun and moon exists vs. a specific door is open.

  Corresponds to probability of change.

- Example: Passing a door

  
  \[
  \begin{align*}
  \text{Open} & \rightarrow \text{open} \\
  \text{open} & \rightarrow \text{can\_pass} \\
  \text{Closed} & \rightarrow \text{closed} \\
  \text{closed} & \rightarrow \text{can\_not\_pass} \\
  \text{can\_pass} & \land \text{can\_not\_pass} \rightarrow \bot \\
  \text{open} & \land \text{closed} \rightarrow \bot
  \end{align*}
  \]
Basic Data Structure

Node

\[ \gamma_{\text{datum}} : \langle \text{datum, label, justifications} \rangle \]

where \text{datum} is send by the problem solver.

- Premise, e.g., \langle p, \{\{}\}, \{()\} \rangle
- Assumption, e.g., \langle A, \{\{A\}\}, \{(A)\} \rangle
- Assumed nodes, e.g., \langle a, \{\{A\}\}, \{(A)\} \rangle
- Derived nodes, e.g., \langle \text{can pass}, \{\{Open\}\}, \{(open)\} \rangle
- Falsity, \langle \bot, \ldots, \ldots \rangle. Inconsistent environments are called NOGOODS.

Logical interpretations of

\[ \langle n, \{\{A_1, \ldots, A_n\}, \{B_1, \ldots, B_m\}, \ldots\}, \{(x_1, \ldots, x_k), (y_1, \ldots, y_j), \ldots\} \rangle \]

\[ (A_1 \land \ldots \land A_n) \lor (B_1 \land \ldots \land B_m) \lor \ldots \rightarrow n \]
\[ (x_1 \land \ldots \land x_k) \lor (y_1 \land \ldots \land y_j) \lor \ldots \rightarrow n \]
ATMS Algorithm (I)

- Central task is do maintain node labels

- Only necessary when justification added

- \( J \) is supplied \( \Rightarrow \) \text{PROPAGATE}(J, \Phi, \{\{\}\}) \) is called. \( \Phi \) indicates the absence of an optional antecedence node.

- Only incremental changes are propagated through the ATMS
ATMS Algorithm (II)

ALGORITHM PROPAGATE \((x_1, \ldots, x_n \rightarrow x_{n+1}), a, I\)

1. [Compute the incremental update] 
   \(L = \text{WEAVE}(a, I, \{x_1, \ldots, x_n\})\). If \(L\) is empty, return.

2. [Update label and recur] UPDATE\((L, x_{n+1})\).

ALGORITHM UPDATE\((L, n)\)

1. [Detect nogoods] If \(n = \perp\) then call NOGOOD\((E)\) on each \(E \in L\) and return \(\{\}\).

2. [Update \(n\)'s label ensuring minimality]
   (a) Delete every environment from \(L\) which is a superset of some label environment of \(n\).
   
   (b) Delete every environment from the label of \(n\) which is a superset of some element of \(L\).
   
   (c) Add every remaining environment of \(L\) to the label of \(n\).

3. [Propagate the incremental change to \(n\)'s label to its consequences] For every justification \(J\) in which \(n\) is mentioned as an antecedent call PROPAGATE\((J, n, L)\).
ATMS Algorithm (III)

ALGORITHM \texttt{WEAVE}(a, I, X)

1. [Termination condition] If $X$ is empty, return $I$.

2. [Iterate over the antecedent nodes] Let $h$ be the first node of the list $X$ and $R$ the rest.

3. [Avoid computing the full label] If $h = a$, return $\texttt{WEAVE}(\emptyset, I, R)$.

4. [Incrementally construct the incremental label] Let $I'$ be the set of all environments formed by computing the union of an environment of $I$ and an environment of $h$'s label.

5. [Ensure that $I'$ is minimal and contains no known inconsistency] Remove from $I'$ all duplicates, nogoods, as well as any environment subsumed by any other.

6. Return $\texttt{WEAVE}(a, I', R)$.

ALGORITHM \texttt{NOGOOD}(E)

1. Mark $E$ as nogood.

2. Remove $E$ and any superset from every node label.
Consider the Coffee Machine Example before adding the fact \textit{no\_coffee}.

\begin{verbatim}
  water   \{\{Water\}\}
  beans   \{\{Beans\}\}
  request \{\{Request\}\}
  coffee  \{\{Water, Beans, Request\}\}
  no\_coffee \{}
  no\_beans \{}
  no\_water \{}
  no\_request \{}
  \bot \{}
\end{verbatim}

And add the fact \textit{no\_coffee} by calling \textbf{PROPAGATE}((→ \textit{no\_coffee}), \Phi, \{\{\}\}).
Example (cont.)

PROPAGATE((→ no\_coffee),Φ,{{}})

\[ L = \text{WEAVE}(Φ,{{}},{{}}) = {{}} \]

UPDATE({{{}}},no\_coffee)

\[ \langle \text{no\_coffee},{{}},\ldots \rangle \]

PROPAGATE((coffee ∧ no\_coffee → ⊥), no\_coffee, {{}})

\[ L = \text{WEAVE}(\text{no\_coffee},{{}},{\text{coffee, no\_coffee}}) \]
\[ h = \text{coffee}, R = \{\text{no\_coffee}\} \]
\[ I' = \{\{\text{Water, Beans, Request}\}\} \]

WEAVE(no\_coffee,

\{\{\text{Water, Beans, Request}\}, \{no\_coffee\}\})
\[ h = \text{no\_coffee}, R = \{\} \]

WEAVE(Φ,{{Water, Beans, Request}}, {{}})

\[ L = \{\{\text{Water, Beans, Request}\}\} \]

UPDATE({{{Water, Beans, Request}}},⊥)

NOGOOD({{Water, Beans, Request}}) (*)

Labels at position (*):

water  {Water}
beans  {Beans}
request  {Request}
coffee  {}
no\_coffee  {}
no\_beans  {}
no\_water  {}
no\_request  {}
⊥  {Water, Beans, Request}
Some other Examples

- Multiple environments

\[
\begin{align*}
A & \rightarrow a & B & \rightarrow b \\
a & \rightarrow c & b & \rightarrow c \\
c, d & \rightarrow \bot & d & \\
\end{align*}
\]

- Multiple environments II

\[
\begin{align*}
A & \rightarrow a & B & \rightarrow b \\
C & \rightarrow c & D & \rightarrow d \\
a, b & \rightarrow e & a, c & \rightarrow e \\
a, d & \rightarrow e & b, c & \rightarrow e \\
b, d & \rightarrow e & c, d & \rightarrow e \\
e, f & \rightarrow \bot & f & \\
\end{align*}
\]
Properties of ATMS

- If there are \( n \) assumptions, then there are potentially \( 2^n \) contexts.

- There are \( \binom{n}{k} \) environments having \( k \) assumptions.

- Label update for the ATMS is NP-complete.

The prove is done by (1) showing that the ATMS is in NP, and (2) find a polynomial reduction from a known NP-hard problem.

ad (1): ATMS must be in NP. Given a particular input, we can guess a set \( S \) of propositions of size \( k - 1 \), set them to TRUE and run the Horn clause deduction in linear time to confirm that no contradiction arises.

ad (2) Reduction from the Max Clique Problem (MCP): Given an instance graph \( G \), and an integer \( k \), we want to find out if \( G \) contains as a subgraph a clique of size \( k - 1 \) or more.
Prove (cont.) ATMS is NP-complete

Polynomial reduction from MCP to ATMS: \( n \) be the number of nodes in \( G \). For every \( v \in G \) let \( y_v \) be a proposition saying \( v \) is in the clique. The \( y_v \)'s are in the set of assumptions \( A \) and propositions \( X \). Formula \( F \) is a conjunction of clauses: For every pairs \( \langle v, w \rangle \) of nodes in \( G \) which are not adjacent, add the rule \( y_v \land y_w \rightarrow \bot \). This means \( v \) and \( w \) does not belong to the same clique.

**Claim** \( G \) contains a clique of size \( k - 1 \) or more iff there exists a set \( S \) of assumptions of size \( k - 1 \), that, if all set to TRUE will leave \( F \) satisfiable.

**Prove (Claim):**

\((\Rightarrow)\) \( G \) contains a clique \( V \) of size \( k - 1 \). Let all \( y_v \in S \) where \( v \in V \) be TRUE and the rest to FALSE. It is trivial to see that no rule in \( F \) fires. Thus, \( F \) is satisfiable.
Prove (cont. (II)) ATMS is NP-complete

$(\Leftarrow) \ S$ is a set of $k - 1$ assumptions that, if all set to TRUE, will leave $F$ satisfiable. Let $V_S$ be the set of corresponding nodes $v$, for which $y_v \in S$. We claim that $V_S$ is a clique. Suppose the converse. Then there must be nodes $v$ and $w$ in $V_S$ that are not adjacent in $G$. But then $y_v \land y_w \rightarrow \bot$ must be in $F$. Hence, $F$ cannot be satisfiable, contradicting our initial assumptions. ■

The ATMS is NP-complete
**Extensions - Hyper-resolution**

**Problem**: Horn clauses cannot encode every propositional formula.

**Solution**: Extend the ATMS to accept positive clauses of assumptions $A_1, \ldots, A_n$.

\[
\text{choose } \{A_1, \ldots, A_n\}
\]

represents

\[
A_1 \lor \ldots \lor A_n
\]

All propositional formulas can be expressed using horn clauses and positive clauses.

The basic ATMS algorithm no longer ensures label consistency or completeness!
Hyper-resolution (II)

Example:

\[
\begin{align*}
\text{choose}\{A, B\} \\
A \land C &\rightarrow \bot \\
B \land C &\rightarrow \bot
\end{align*}
\]

The basic ATMS algorithm does not find the nogood \{C\}. It does find \{A, C\} and \{B, C\}!

Hyper-resolution Rule:

\[
\begin{align*}
\text{choose}\{A_1, \ldots, A_n\} \\
\text{nogood } &\alpha_i \text{ where } A_i \in \alpha_i \text{ and } A_j \notin \alpha_i, i \neq j, \text{ for all } 1 \leq i, j \leq n \\
\text{nogood } &\bigcup_i [\alpha_i \setminus \{A_i\}]
\end{align*}
\]

Example (cont.):

\[
\begin{align*}
\text{choose}\{A, B\} \\
\text{nogood}\{A, C\} &\quad A \lor B \\
\text{nogood}\{B, C\} &\quad \neg A \lor \neg C \\
\text{nogood}\{C\} &\quad \neg B \lor \neg C \\
\text{nogood}\{C\} &\quad \neg C
\end{align*}
\]
The NATMS

Negated Assumptions ATMS (NATMS) allows negated assumption in the antecedents of justifications.

- Label consistency
- No hyper-resolution rule needed
- Produces more complete node labels
- Better encoding
- The negation of assumption $A$ is a non-assumption node ($\neg A$).
- Choose can be represented by the NATMS. For example

$$\text{choose}\{A, B, C\}$$

is expressed by

$$\neg A \land \neg B \land \neg C \rightarrow \bot.$$
• Observation: Any negative clause of size $k$ is equivalent to any of $k$ implications.

\[ \neg A \lor \neg B \lor \neg C \]

is equivalent to any of:

\[ A \land B \rightarrow \neg C \]
\[ A \land C \rightarrow \neg B \]
\[ B \land C \rightarrow \neg A \]

• NATMS has new inference rule:

\[
\frac{nogood\{A_1, \ldots, A_n, A_{n+1}\}}{A_1, \ldots, A_n \rightarrow \neg A_{n+1}}
\]
NATMS Algorithm (II)

Example: The NATMS discovers new nogood

\[ \text{nogood}\{A, B, C\} \]

and produces the following labels:

\[
\langle \neg A, \{\{B, C\}\} \rangle \\
\langle \neg B, \{\{A, C\}\} \rangle \\
\langle \neg C, \{\{A, B\}\} \rangle 
\]

representing the following justifications

\[
B \land C \rightarrow \neg A \\
A \land C \rightarrow \neg B \\
A \land B \rightarrow \neg C 
\]

Note, it is not necessary to really install the justifications.
The basic algorithm remains except the following.

**ALGORITHM NOGOOD’**(\(E\))

3. [Handle negated assumptions] For every \(A \in E\) for which \(\neg A\) appears in some justification call \(\text{UPDATE}(\{E \setminus \{A\}\}, \neg A)\).

Example:

\[
\begin{align*}
\text{choose}\{A, B\} \text{ represented by:} \\
\neg A \land \neg B & \rightarrow \bot \\
A \land C & \rightarrow \bot \\
B \land C & \rightarrow \bot
\end{align*}
\]

produces 2 nogoods \(\{A, C\}\) and \(\{B, C\}\).

\[
\begin{align*}
\langle \neg A, \{C\} \rangle \\
\langle \neg B, \{C\} \rangle
\end{align*}
\]

which when propagated to \(\neg A \land \neg B \rightarrow \bot\) produces the nogood \(\{C\}\).
Completeness of the NATMS?

The NATMS algorithm ensures label soundness, consistency, minimality but NOT completeness.

Example:

\[
A \rightarrow b \\
\neg A \rightarrow b
\]

Assuming no other justifications the NAMTS computes the label \( \langle b, \{\{A\}\} \rangle \) which is incomplete! \( b \) holds universally.

In most cases completeness not necessary \( \Rightarrow \) therefore omitted in the algorithm.
Encoding Tricks

- [Negated non-assumptions] For every negated non-assumption node $n$ appearing in the antecedents of a justification define a new Assumption $A$ and add two justifications:

$$A \rightarrow n$$
$$\neg A \rightarrow \neg n$$

Example:

$$\neg a \land B \rightarrow c$$
$$a \land D \rightarrow \bot$$

The encoding provides $\langle c, \{\{B, D\}\} \rangle$.

- [Negated assumptions as assumptions] Assume an assumption $A$. $\neg A$ is not seen as assumption. Create new assumption $\sqrt{A}$ which should be the negated $A$. The following justifications must be added:

$$A \land \sqrt{A} \rightarrow \bot$$
$$\neg A \land \neg \sqrt{A} \rightarrow \bot$$

Now $\sqrt{A}$ appears in the labels (while $\neg A$ doesn’t).
Other Extensions

- **Focusing the ATMS**
  
  Avoid label explosion
  
  - Restrict labels to subsets of a focus set
  
  - Restrict labels to an element of a fixed set of environments

- **Integrating probability into the ATMS**
  
  - Dempster-Shafer theory
  
  - Possibilistic theory
  
  - Certainty factors
  
  - Fuzzy Logic
Possibilistic ATMS ($\Pi$-ATMS)

Possibilistic Logic (Dubois and Prade)

Logical sentences = conjunctions of possibilistic propositional clauses.

- **Possibility measure** $\Pi \in [0, 1]$: 
  1. $\Pi(\bot) = 0$, $\Pi(\top) = 1$
  2. $\forall p, \forall q, \Pi(p \lor q) = \max \Pi(p), \Pi(q)$
  3. but $\Pi(p \land q) \leq \min \Pi(p), \Pi(q)$

- **Necessity measure** $N \in [0, 1]$: 
  1. $N(p) = 1 - \Pi(\neg p)$
  2. it follows $\forall p, \forall q, N(p \land q) = \min N(p), N(q)$
  3. and $N(p \lor q) \geq \max N(p), N(q)$

$\Pi$ and $N$ are dual
\(\Pi\)-ATMS: Possibilistic Logic

- \(N(p) = 1\) means that, given the available knowledge, \(p\) is certainly true.

- \(1 > N(p) > 0\) means that, \(p\) is somewhat certain and \(\neg p\) not certain at all.

- \(N(p) = N(\neg p) = 0 (= \Pi(p) = \Pi(\neg p) = 1)\) is the case of total ignorance. Nothing is known about the truth value of \(p\).

- \(0 < \Pi(p) < 1 (= 1 > N(p) > 0)\) means that \(p\) is somewhat impossible.

- \(\Pi(p) = 0\) means that \(p\) is certainly false.


\[ \Pi\text{-ATMS: Possibilistic Logic} \]

- Clause attached with a lower bound of its necessity measure

\[ (f \alpha) \text{ where } \alpha \in [0, 1], N(f) \geq \alpha \]

- Resolution rule

\[ \frac{(c \alpha) \ (c' \beta)}{(\text{Resolvent}(c, c') \ \text{min } \alpha, \beta)} \]

- Example:

C1 \ (\neg a \lor \neg b \lor \neg c \ 0.7)
C2 \ (\neg d \lor c \ 0.4)

From C1 and C2 the clause \((\neg a \lor \neg b \lor \neg d \ 0.4)\) can be derived.
\( \Pi \text{-ATMS: Principles} \)

- Each clause has a weight, i.e., the lower bound of its necessity degree.

- Assumptions may also be weighted.

- \( \Pi \text{-ATMS} \) should answer the following:
  - Under what configuration of assumptions is the proposition \( p \) certain to a degree \( \alpha \)?
  - What is the inconsistency degree of a given configuration of assumptions?
  - In a given configuration of assumption, to what degree is each proposition certain?

- Note, the \( \Pi \text{-ATMS} \) in its original form is more general than the NATMS.
\[ \Pi \text{-ATMS: Definitions} \]

- \([\text{Environment}] [E \alpha] \) is an environment of the proposition \( p \) iff \( N(p) \geq \alpha \) is a logical consequence of \( E \cup Th \) when all assumptions in \( E \) are set to TRUE with degree 1.

- \([\alpha\text{-Environment}] [E \alpha] \) is an \( \alpha \)-environment of \( p \) iff \([E \alpha] \) is an environment of \( p \) and \( \forall \alpha' > \alpha, [E \alpha'] \) is not an environment of \( p \).

- \([\alpha\text{-Nogood}] [E \alpha] \) is a \( \alpha \)-nogood iff \( E \cup Th \) is \( \alpha \)-inconsistent, i.e., \( E \cup Th \models (\bot \alpha) \). A \( \alpha \)-nogood is minimal if there is no other nogood \([E', \beta] \) such that \( E \subset E' \) and \( \alpha \leq \beta \).
\[\Pi\text{-ATMS: Definitions (II)}\]

**Labels** (only using non-weighted assumptions)

- **[(weak) consistency]** \(\forall [E_i \; \alpha_i] \in L(p), \ E_i \cup Th\) is \(\beta\)-inconsistent with \(\beta < \alpha_i\). \(\beta\) ensures that only formulas with weights \(> \beta\), and from which \(p\) can be deduced, are member of the \(p\)'s label.

- **[soundness]** \(L(p)\) is sound iff \(\forall [E_i \; \alpha_i] \in L(p)\) we have \(E_i \cup Th \models (p \; \alpha_i)\).

- **[completeness]** \(L(p)\) is complete iff for every environment \(E'\) such that \(E' \cup Th \models (p \; \alpha')\) then \(\exists [E_i \; \alpha_i] \in L(p)\) such that \(E_i \subseteq E\) and \(\alpha_i \geq \alpha'\).

- **[minimality]** \(L(p)\) is minimal iff it does not contain two environments \([E, \alpha], [E', \alpha']\) such that \(E \subseteq E'\) and \(\alpha \geq \alpha'\).
\(\Pi\text{-ATMS: Remarks}\)

- Inconsistent environments can be element of a node label.

- Subset minimality of labels is not required.

- Solutions can be ranked.
Bibliography


