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

$$\begin{array}{ll} a, & c, \\ a \to b, & c \to d, \\ a \wedge c \to e, & b \wedge d \to \bot. \end{array}$$

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

Theory Th is inconsistent!

$$\begin{array}{rcl} a, a \to b & \models & b \\ c, c \to d & \models & d \\ a, c, a \wedge c \to e & \models & e \\ b, d, b \wedge d \to \bot & \models & \bot \end{array}$$

From $\bot$ follows everything!

Example (II)

Aim: Eliminate inconsistency!

Default Logic (Only normal defaults):

$$\begin{array}{ll} \frac{true \quad A}{A}, & \frac{true \quad C}{C} \\ A \rightarrow b, & C \rightarrow d \\ A \wedge C \rightarrow e, & b \wedge d \rightarrow \bot. \end{array}$$

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 . . . Consistent set of defaults (ATMS assumptions)

$$\begin{array}{lll}
1. & A & \langle A, \{\{A\}\}\rangle \\
2. & C & \langle C, \{\{C\}\}\rangle \\
3. & A \to b & \langle b, \{\{A\}\}\rangle \\
4. & C \to d & \langle d, \{\{C\}\}\rangle \\
5. & A \wedge C \to e & \langle e, \{\{A, C\}\}\rangle \\
6. & b \wedge d \to \perp & \langle \perp, \{\{A, C\}\}\rangle \\
 & & \langle e, \{\}\rangle
\end{array}$$

e is no longer supported! b is supported by A and d is supported by C.

Simple TMS System

- Algorithm

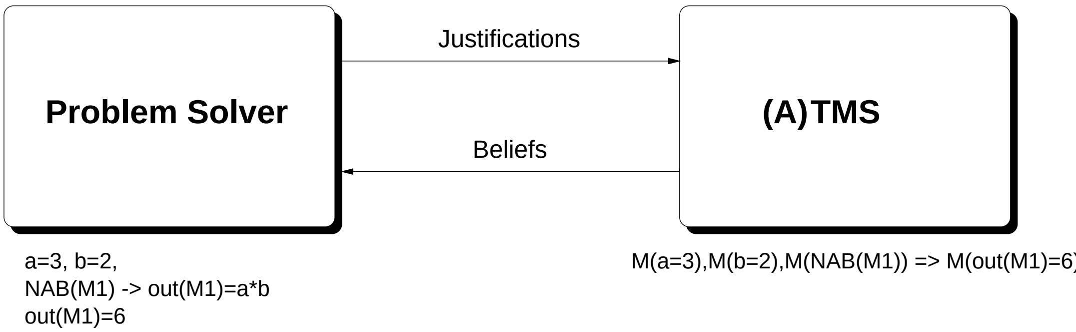
  1. Let Th be a set of facts and rules.
  2. If Th is consistent exit the algorithm.
  3. Otherwise, select a fact or rule r from Th.
  4. Remove r from Th, i.e., $Th = Th \setminus \{r\}$, and goto step 2.

- 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 $m1$ with behavior $\neg ab(m1) \to out(m1) = in_1(m1) \cdot in_2(m1)$

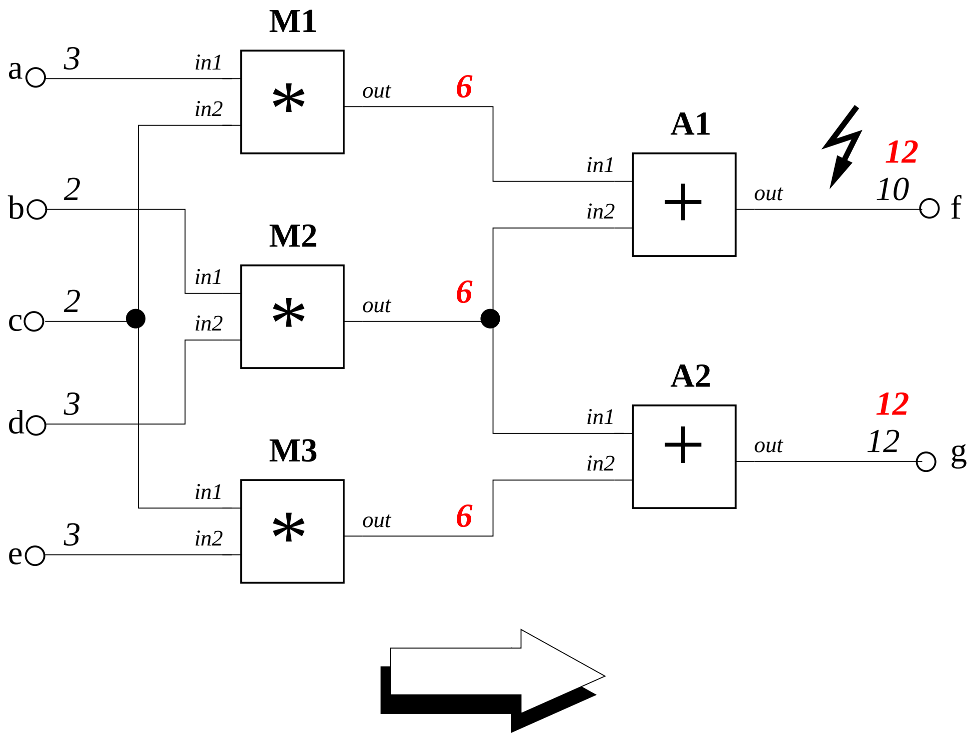
Problem solver knows that $in_1(m1) = 3$, $in_2(m1) = 3$, and the behavior. Under the assumption that $\neg ab(m1)$ the problem solver can conclude $out(m1) = 6$.

Problem solver sends justification to ATMS:

$$\Gamma(in_1(m1) = 3) \wedge \Gamma(in_2(m1) = 2) \wedge \Gamma(\neg ab(m1)) \to \Gamma(out(m1) = 6)$$

The data $\Gamma(\neg ab(m1))$ is an assumption.

Communication with ATMS (D74 Ex.)



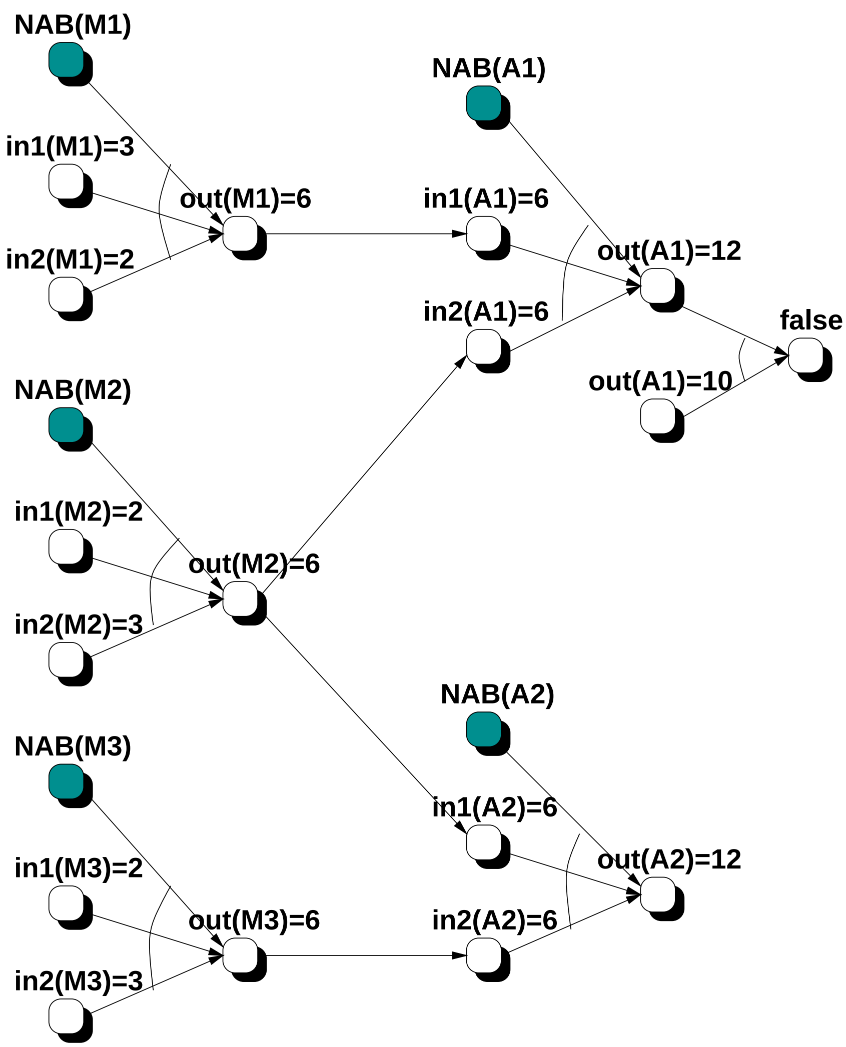
Behavior and structure $mult(C) \wedge \neg ab(C) \rightarrow out(C) = in_1(C) * in_2(C)$, $plus(C) \wedge \neg ab(C) \rightarrow out(C) = in_1(C) + in_2(C)$, $mult(M1)$, $mult(M2)$, $mult(M3)$, $plus(A1)$, $plus(A2)$, $in_1(M1) = a$, $in_2(M1) = c$, ...

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 $NAB(M1), in_1(M1) = 2, in_2(M1) = 3 \rightarrow out(M1) = 6$, $NAB(M1), out(M1) = 6, in_1(M1) = 2 \rightarrow in_2(M1) = 3$, ...

ATMS - Data as Graph (D74 Example)

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 ($\perp$) can be derived, i.e., $E \cup Th \models \perp$.

- 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.

$$E \cup Th \models n$$

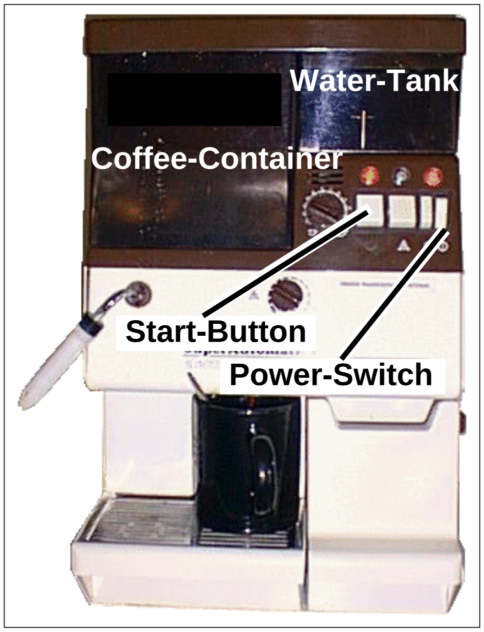
Complete A label L for node n is complete iff every consistent environment $E \notin L$ for which $E \cup Th \models n$ is a superset of some $E' in L$, i.e, $E' \subset E$.

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 $E' \cup Th \models n$.

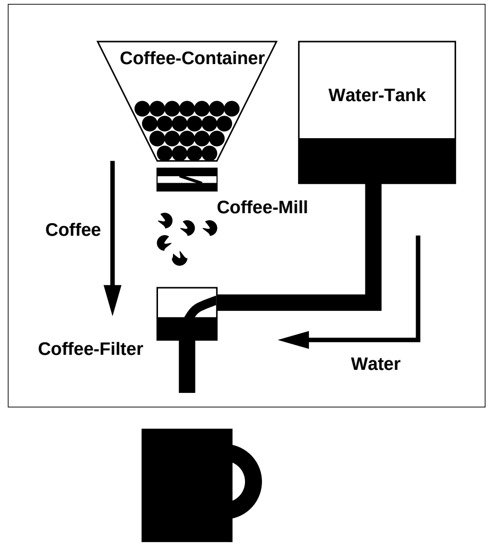
Consequences

- **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



Coffee-Machine



Model

$$
\begin{gathered}
Request \rightarrow request \\
Water \rightarrow water \\
Beans \rightarrow beans \\
request \wedge water \wedge beans \rightarrow coffee \\
coffee \rightarrow request \\
coffee \rightarrow water \\
coffee \rightarrow beans
\end{gathered}
$$

Coffee Machine (II)

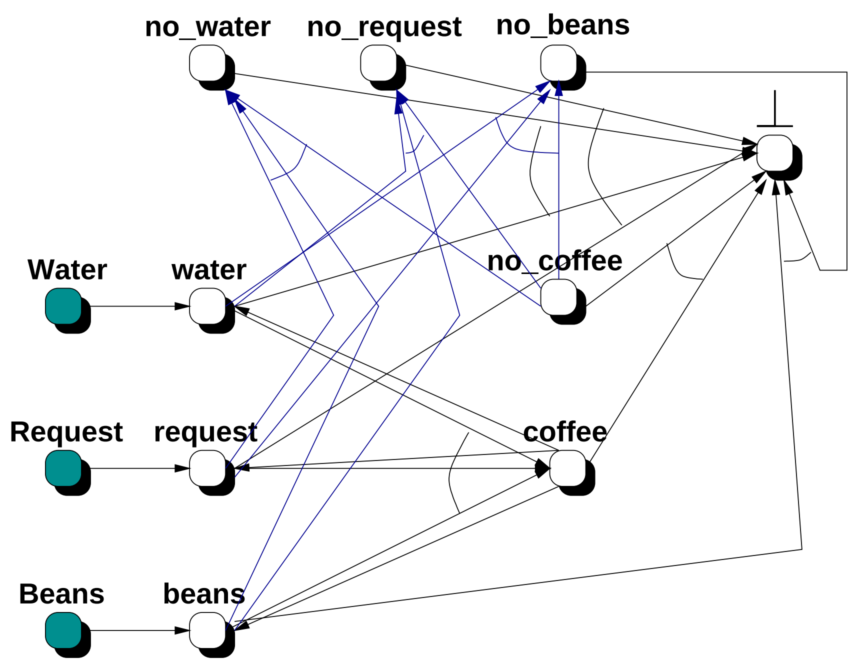
Model (cont.)

$$\frac{\begin{array}{c} no\_coffee \wedge request \wedge water \rightarrow no\_beans \\ no\_coffee \wedge request \wedge beans \rightarrow no\_water \\ no\_coffee \wedge water \wedge beans \rightarrow no\_request \end{array}}{\begin{array}{c} beans \wedge no\_beans \rightarrow \perp \\ request \wedge no\_request \rightarrow \perp \\ water \wedge no\_water \rightarrow \perp \\ coffee \wedge no\_coffee \rightarrow \perp \end{array}}$$

Observations

$$no\_coffee$$

Coffee Machine (III)



water	{{Water}}
beans	{{Beans}}
request	{{Request}}
coffee	{}
no_coffee	{{}}
no_beans	{{Water,Request}}
no_water	{{Beans,Request}}
no_request	{{Water,Beans}}
$\perp$	{{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$\}\}$
$\perp$	$\{\{$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

$$
\begin{gathered}
Open \to open \\
open \to can\_pass \\
Closed \to closed \\
closed \to can\_not\_pass \\
can\_pass \wedge can\_not\_pass \to \bot \\
open \wedge closed \to \bot
\end{gathered}
$$

Basic Data Structure

Node

$$\gamma_{datum} : \langle datum, label, justificiations \rangle$$

where $datum$ 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., $\langle can\_pass, \{\{Open\}\}, \{(open)\}\rangle$
- Falsity, $\langle \bot, \ldots, \ldots \rangle$. Inconsistent environments are called NOGOODS.

Logical interpretations of

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

$$(A_1 \wedge \ldots \wedge A_n) \vee (B_1 \wedge \ldots \wedge B_m) \vee \ldots \rightarrow n \\ (x_1 \wedge \ldots \wedge x_k) \vee (y_1 \wedge \ldots \wedge y_j) \vee \ldots \rightarrow n$$

ATMS Algorithm (I)

- Central task is do maintain node labels
- Only necessary when justification added
- J is supplied $\Rightarrow$ **PROPAGATE**(J, Φ, $\{\{\}\}$) is called. Φ 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 = **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 justificiation J in which n is mentioned as an antecedent call **PROPAGATE**(J,n,L).

ATMS Algorithm (III)

ALGORITHM **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 **WEAVE**(Φ,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 **WEAVE**(a,I',R).

ALGORITHM **NOGOOD**(E)

1. Mark E as nogood.

2. Remove E and any superset from every node label.

Example

Consider the Coffee Machine Example before adding the fact no_coffee.

water	$\{\{$Water$\}\}$
beans	$\{\{$Beans$\}\}$
request	$\{\{$Request$\}\}$
coffee	$\{\{$Water, Beans, Request$\}\}$
no_coffee	$\{\}$
no_beans	$\{\}$
no_water	$\{\}$
no_request	$\{\}$
$\perp$	$\{\}$

And add the fact no_coffee by calling

PROPAGATE(($\rightarrow no\_coffee$),Φ,$\{\{\}\}$).

Example (cont.)

PROPAGATE(($\rightarrow no\_coffee$),Φ,$\{\{\}\}$)
  L = **WEAVE**(Φ,$\{\{\}\}$,$\{\}$) = $\{\{\}\}$
  UPDATE($\{\{\}\}$,no_coffee)
    $\langle no\_coffee, \{\{\}\}, \ldots \rangle$
    PROPAGATE(($coffee \wedge no\_coffee \rightarrow \perp$), no_coffee, $\{\{\}\}$)
      L = **WEAVE**(no_coffee,$\{\{\}\}$,$\{coffee, no\_coffee\}$)
        $h = coffee$, $R = \{no\_coffee\}$
        $I' = \{\{Water, Beans, Request\}\}$
        WEAVE(no_coffee,
          $\{\{Water, Beans, Request\}\}$, $\{no\_coffee\}$)
          $h = no\_coffee$, $R = \{\}$
          WEAVE(Φ,$\{\{Water, Beans, Request\}\}$, $\{\}$)
      $L = \{\{Water, Beans, Request\}\}$
      UPDATE($\{\{Water, Beans, Request\}\}$,$\perp$)
        NOGOOD($\{Water, Beans, Request\}$) **(*)**
    ⋮

Labels at position (*):

water	$\{\{$Water$\}\}$
beans	$\{\{$Beans$\}\}$
request	$\{\{$Request$\}\}$
coffee	$\{\}$
no_coffee	$\{\{\}\}$
no_beans	$\{\}$
no_water	$\{\}$
no_request	$\{\}$
$\perp$	$\{\{$Water, Beans, Request$\}\}$

Some other Examples

- Multiple environments

$$\begin{array}{ll} A \to a & B \to b \\ a \to c & b \to c \\ c, d \to \bot & d \end{array}$$

- Multiple environments II

$$\begin{array}{ll} A \to a & B \to b \\ C \to c & D \to d \\ a, b \to e & a, c \to e \\ a, d \to e & b, c \to e \\ b, d \to e & c, d \to e \\ e, f \to \bot & f \end{array}$$

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 $y_v \wedge y_w \rightarrow \perp$. 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 $y_v \wedge y_w \rightarrow \perp$ must be in F. Hence, F cannot be satisfiable, contradicting our initial assumptions. $\blacksquare$

The ATMS is NP-complete

Extensions - Hyper-resolution

Problem: Horn clauses cannot encode every propositional formula.

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

$$choose\,\{A_1, \ldots, A_n\}$$

represents

$$A_1 \vee \ldots \vee A_n$$

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:

$$\begin{array}{c} choose\{A, B\} \\ A \wedge C \rightarrow \bot \\ B \wedge C \rightarrow \bot \end{array}$$

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

Hyper-resolution Rule:

$$\frac{\begin{array}{l} choose\{A_1, \ldots, A_n\} \\ nogood \;\; \alpha_i \text{ where } A_i \in \alpha_i \text{ and } A_j \notin \alpha_i, i \neq j, \text{ for all } 1 \leq i, j \leq n \end{array}}{nogood \bigcup_i [\alpha_i \setminus \{A_i\}]}$$

Example (cont.):

$$\frac{\begin{array}{l} choose\{A, B\} \\ nogood\{A, C\} \\ nogood\{B, C\} \end{array}}{nogood\{C\}} \qquad \frac{\begin{array}{l} A \vee B \\ \neg A \vee \neg C \\ \neg B \vee \neg C \end{array}}{\neg C}$$

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

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

  is expressed by

  $$\neg A \wedge \neg B \wedge \neg C \rightarrow \perp .$$

NATMS Algorithm

- Observation: Any negative clause of size k is equivalent to any of k implications.

$$\neg A \vee \neg B \vee \neg C$$

is equivalent to any of:

$$\begin{array}{c} A \wedge B \rightarrow \neg C \\ A \wedge C \rightarrow \neg B \\ B \wedge C \rightarrow \neg A \end{array}$$

- 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

$$nogood\{A, B, C\}$$

and produces the following labels:

$$\begin{array}{c}\langle \neg A, \{\{B, C\}\}\rangle \\ \langle \neg B, \{\{A, C\}\}\rangle \\ \langle \neg C, \{\{A, B\}\}\rangle\end{array}$$

representing the following justifications

$$\begin{array}{c}B \wedge C \rightarrow \neg A \\ A \wedge C \rightarrow \neg B \\ A \wedge B \rightarrow \neg C\end{array}$$

Note, it is not necessary to really install the justifications.

NATMS Algorithm (III)

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 **UPDATE**($\{E \setminus \{A\}\}$, $\neg A$).

Example:

$$\begin{array}{c} choose\{A, B\} \text{ represented by:} \\ \neg A \wedge \neg B \rightarrow \bot \\ A \wedge C \rightarrow \bot \\ B \wedge C \rightarrow \bot \end{array}$$

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

$$\begin{array}{c} \langle \neg A, \{C\} \rangle \\ \langle \neg B, \{C\} \rangle \end{array}$$

which when propagated to $\neg A \wedge \neg B \rightarrow \bot$ 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:

$$\neg a \wedge B \rightarrow c$$
$$a \wedge D \rightarrow \perp$$

  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 $\surd A$ which should be the negated A. The following justifications must be added:

$$A \wedge \surd A \rightarrow \perp$$
$$\neg A \wedge \neg \surd A \rightarrow \perp$$

  Now $\surd A$ 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 (Π-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. $\forall p, \forall q, \Pi(p \vee q) = \max \Pi(p), \Pi(q)$
  3. but $\Pi(p \wedge q) \leq \min \Pi(p), \Pi(q)$
- Necessity measure $N \in [0, 1]$:
  1. $N(p) = 1 - \Pi(\neg p)$
  2. it follows $\forall p, \forall q, N(p \wedge q) = \min N(p), N(q)$
  3. and $N(p \vee q) \geq \max N(p), N(q)$

Π and N are dual

Π-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.

Π-ATMS: Possibilistic Logic

- Clause attached with a lower bound of its necessity measure

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

- Resolution rule

$$\frac{(c \quad \alpha) \qquad (c' \quad \beta)}{(\text{Resolvent}(c, c') \quad \min \alpha, \beta)}$$

- Example:

C1	$(\neg a \vee \neg b \vee \neg c \quad 0.7)$
C2	$(\neg d \vee c \quad 0.4)$

From C1 and C2 the clause $(\neg a \vee \neg b \vee \neg d \quad 0.4)$ can be derived.

Π-ATMS: Principles

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

- Assumptions may also be weighted.

- A Π-ATMS should answer the following:
  - Under what configuration of assumptions is the proposition p certain to a degree α?
  - 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 Π-ATMS in its original form is more general than the NATMS.

Π-ATMS: Definitions

- [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.

- [α-Environment] $[E \;\; \alpha]$ is an α-environment of p iff $[E \;\; \alpha]$ is an environment of p and $\forall \alpha' > \alpha, [E \;\; \alpha']$ is not an environment of p.

- [α-Nogood] $[E \;\; \alpha]$ is a α-nogood iff $E \cup Th$ is α-inconsistent, i.e., $E \cup Th \models (\perp \alpha)$. A α-nogood is minimal if there is no other nogood $[E', \beta]$ such that $E \subset E'$ and $\alpha \leq \beta$

Π-ATMS: Definitions (II)

Labels (only using non-weighted assumptions)

- [(weak) consistency] $\forall\,[E_i\;\;\alpha_i] \in L(p)$, $E_i \cup Th$ is β-inconsistent with $\beta < \alpha_i$. β 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 $E_i \subset E$ and $\alpha_i \geq \alpha'$.

- [minimality] $L(p)$ is minimal iff it does not contain two environments $[E, \alpha]$, $[E', \alpha']$ such that $E \subset E'$ and $\alpha \geq \alpha'$.

Π-ATMS: Remarks

- Inconsistent environments can be element of a node label.
- Subset minimality of labels is not required.
- Solutions can be ranked.

Bibliography

kle86 Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

kle86d Johan de Kleer. Extending the ATMS. *Artificial Intelligence*, 28:163–196, 1986.

kle86c Johan de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28:197–224, 1986.

kle88 Johan de Kleer. A general labeling algorithm for assumption-based truth maintenance. In *Proceedings AAAI*, pages 188–192, Saint Paul, Minnesota, August 1988. Morgan Kaufmann.

for88 Kenneth D. Forbus and Johan de Kleer. Focusing the ATMS. In *Proceedings AAAI*, pages 193–198, Saint Paul, Minnesota, August 1988. Morgan Kaufmann.

ruten91 Vladislav Rutenburg. Complexity classification of truth maintenance systems. In *STACS*, pages 372 – 382, 1991.