

An Overview of Non-monotonic Reasoning and Reason Maintenance Contents

1. Introduction
2. The Importance of Belief Revision
3. Non-monotonic Reasoning
4. Belief Revision Techniques
5. RMS as Data Dependency Network Management Systems
6. Justification-Based Reason Maintenance
7. Assumption-Based Reason Maintenance
8. Some Actual Problems for RMS

Viewpoint: Problem Solving

Revise appropriately the totality of beliefs held by a problem solver when facing any of the following situations:

- New information from the outside world has to be assimilated
- Premises turn out to be erroneous or change over time
- Assumptions made during problem solving are violated
- An obvious contradiction is encountered
- etc.

Reason Maintenance Systems (RMS) are computational tools to support approaches to the belief revision problem.

The Importance of Belief Revision

The need for belief revision may arise in a variety of situations. Some examples:

- Representing dynamic worlds
Example: If you pick up your red cup which is on a saucer which is on the table it is still red, but no longer on the saucer.
Frame problem: Identify efficiently and update exactly those beliefs affected by a change.
- Time varying input
Example: A system may be supplied with dynamic information about facts (e.g. measurements of a certain quantity like a patient's pulse rate) that in reality did not change.
Only derivations based on the retracted input have to be abandoned.

- Contradictions
Example: The problem solver derives both that the cup is on the saucer which is on the table and that the cup is in the cupboard.
Have to identify the source of conflict and withdraw any belief depending on conflicting assumptions or inconsistent data.
- Non-monotonic inference
In many cases the knowledge available only suffices to formulate general rules that usually apply but also allow for exceptions.
Example: If you want to cross the river then use your boat unless something is evidently wrong with the boat.
Qualification problem: Jumping to tentative conclusions until there is contradictory evidence.
“unless something is evidently wrong” will often be replaced by “unless you have shown... / unless you can show with some minor effort”.
Have to provide for belief revision not only when additional premises may be added but also when reasoning from a fixed premise set.

Non-monotonic Reasoning

Classical logic: If a formula p follows from a set of premises Q then p also follows from any superset of Q (monotonicity property). As your set of beliefs grows, so does the set of conclusions that can be drawn.

▷ Commonsense reasoning in general is not monotonic.

The famous *Tweety* example:

- If we know that *Tweety* is a bird, we conclude that he can fly since birds typically fly.
- Given the information that *Tweety* is a penguin we certainly withdraw our former conclusion without withdrawing any of our former premises:
We still believe that *Tweety* is a bird and that birds typically fly.

Uses of Non-monotonic Reasoning

- Inheritance hierarchies
“Absence of information to the contrary” makes an inference non-monotonic, or defeasible.
Inheriting properties by default. (*Tweety* . . .)
- Closed-world databases
Closed world assumption: If a certain fact is not contained in the database (or cannot be derived), the default conclusion is drawn that the fact does not hold.
Negation is treated as a “failure to prove” (PLANNER, PROLOG)

- Diagnosis

Whenever we use a piece of equipment we do so without being able to prove that it will work. If the device does not work, this default assumption is violated.

We then can examine the symptoms, and (in many cases) find out which of our assumptions about the functionality of the internal components was in error: failure diagnosis based on an understanding of the design of a device.

- Reasoning about action

- Qualification problem: Whenever an action is performed, there are may “hidden” preconditions that would prevent the action from succeeding. We assume, in a non-monotonic way, that they are not relevant to the particular action in question.

- Frame problem: We assume that actions change only what they are known to change (...frame axioms)

- * Consistency view: Anything true before the action will be true afterward, provided that this is consistent with the effects of the action.
- * Minimization view: As little as possible changes when an action occurs.

- Other applications

- Vision, e.g. default conclusions about the rest of a scene we can't see.
- Natural language understanding, e.g. default conclusions about presuppositions, or or how the sentence or story or dialogue will continue.
- Temporal logic, e.g. non-monotonic solutions to the frame problem.

Non-Monotonic Reasoning

- ▷ The current trend towards substantial real world applications urges for a deeper understanding of the theoretical foundations of non-monotonic reasoning.

Formal approaches

1. Draw default conclusions in absence of information to the contrary (e.g. inheritance hierarchies, reasoning about action).
2. Minimize the extent of some predicate or relation (e.g. closed-world databases, diagnosis).

● Ad 1.: Proof-based approaches

- Default logic (Reiter)

Default rules: not part of the logical language as such, but rules of inference

$$\frac{\alpha : \beta}{\gamma}$$

α : precondition (Tweety is a bird)

β : clause to be checked for consistency with the database

(Tweety can fly)

γ : Clause added to the database if β is consistent (Tweety can fly)

$\beta = \gamma$: normal default rule

- Modal approaches

McDermott/Doyle's non-monotonic logic

Moore's autoepistemic logic

Belief Revision Techniques

The capability of appropriately revising its current set of beliefs whenever needed is crucial for any non-monotonic reasoning system.

Issues to be addressed:

- Ad 2.: Minimization-based approaches
 - Circumscription (McCarthy)
Describes default rules in terms of "abnormality"
("abnormal" birds don't fly)
- Objections:
 - "Non-monotonic logic" is a contradiction in terms (Israel)
 - Formal approaches to non-monotonic inference are non-semidecidable (vs. entailment in classical logic), i.e. large effort for consistency checking required, mistakes are possible

- Inference problem: deriving new beliefs from existing ones
- Disbelief propagation: identifying beliefs which become arguable when other beliefs have changed
- Revision problem: resolving contradictions by minimal mutilation of the set of beliefs
- Nonmonotonicity problem: dealing with non-monotonically justified beliefs — beliefs which depend on the disbelief of something else

Historical Approaches

- Operator application to pattern-indexed data bases. Modify database by using parameterized operators that add and erase entries (e.g. STRIPS)
- Change-triggered data base demons. “Demon” procedures monitor the database and react to certain conditions on the database, specifically to the addition or erasure of an assertion.

Explicit Data Dependencies

“Every dubitable assertion in the data base needs to have explicitly recorded exactly which rule and assertions were used in its derivations.” (Hewitt 1975)

▷ The key idea behind any Reason Maintenance System (RMS)

In the case of non-monotonic rules, the derivation of a conclusion may depend not only on the presence of some beliefs but also on the absence of others.

Advantages:

- Dependency directed backtracking: Having explicit dependency information available, a consistency maintenance routine that handles contradictions can trace back the dependency links to find the assumptions and premises underlying a conflict.
- Saving rederivations (“cache”): Keeping “believed” and “disbelieved” status of assertions.
- Explanation facilities: Derivation history, NOT domain principles.

RMS as Data Dependency Network Management Systems

- Base belief revision algorithms on explicit data dependencies and
- separate them from the problem specific application algorithms.
- ▷ RMS plays the role of a data management system interacting with a superior problem solver through some predefined scheme.
- Principal RMS (abstract) data structure:
Data dependency network (DDN) composed of assertions and justifications.

DDN: Directed graph $G = (V, E)$

Vertices V :

- nodes N to denote propositions
- Justifications J to denote sets of justifications used in a derivation, i.e. representation of derivation steps
- $N \cap J = \emptyset$

Edges E : point from $j \in J$ to $n \in N$ or from n to j , i.e.

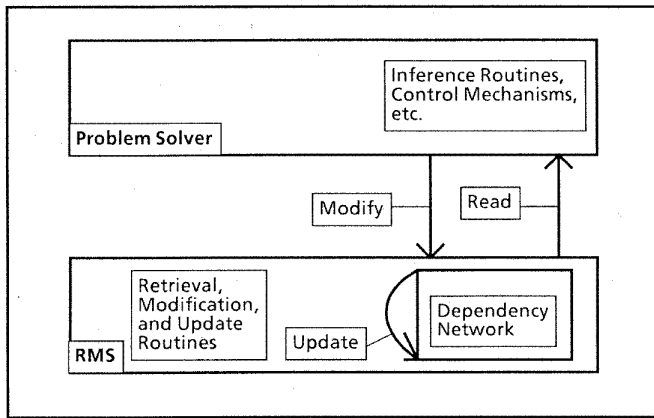
$$E \subseteq (N \times J) \cup (J \times N)$$

j supports n if there is a link from j to n

n participates in j if there is a link from n to j

premise justification: justification without incoming edges

premise: node supported by a premise justification

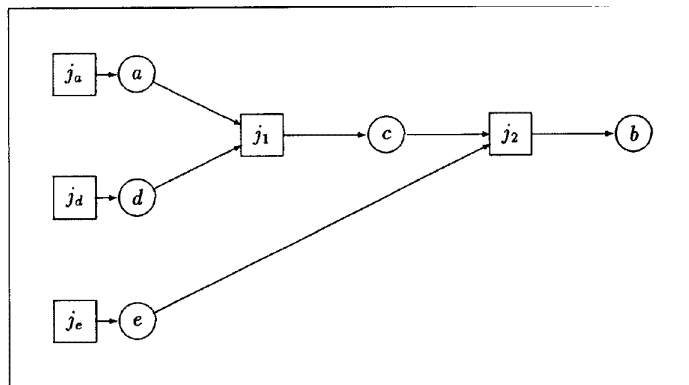


Monotonic DDNs

Example:

- $a \equiv \text{Man}(\text{FRED})$
- $b \equiv \text{Person}(\text{FRED})$
- $c \equiv \text{Human}(\text{FRED})$
- $d \equiv (\forall x : \text{Man}(x) \Rightarrow \text{Human}(x))$
- $e \equiv (\forall x : \text{Human}(x) \Rightarrow \text{Person}(x))$
- $f \equiv (\forall x : \text{Person}(x) \Rightarrow \text{Human}(x))$

A simple Data Dependency Network



Assume set of premises $\{a, d, e\}$,
conclude set of believed propositions $\{a, b, c, d, e\}$

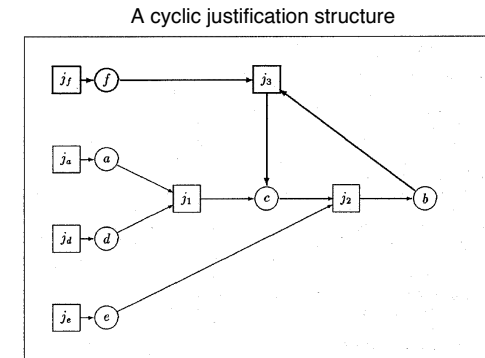
- Procedure to remove derived beliefs in a DDN which are no longer believed:
 1. If a justification j is removed, then check whether the nodes $\{n_j\}$ which the justification supported have another justification. Remove all nodes which do not have an alternate justification.
 2. Remove the justifications which the removed nodes pointed to.
 3. If in the previous step a justification got removed, call the entire procedure recursively. (For the sake of efficiency, we do not have to remove the nodes and justifications, but instead could merely mark them as currently unbelievied.)

Example: Assume we lose confidence in the fact that every Human is a Person, i.e. we want to retract proposition e , hence delete premise justification j_e . Since e is no longer justified, e is deleted, and consequently, j_2 . The recursive call then removes b .

In the worst case, the algorithm performs linearly in the number of vertices of the DDN (average case: much smaller).

Problem: The algorithm may fail to remove not well-founded beliefs — beliefs that cannot be derived any longer from the set of premises.

Example: Add proposition f that every Person is a Human.



Removing the premise a that Fred is a Man would correctly remove justification j_1 , but the derived beliefs b and c would not be touched because both are still justified.

Solution: Current support strategy → modification of the algorithm

Non-monotonic DDNs

- Monotonic DDNs: all nodes participating in a justification are believed nodes.
- Non-monotonic DDNs: take also disbeliefs into account.
- Distinguish among the incoming links of justifications:
 - links which come from nodes which have to be believed (IN-set)
 - links which come from nodes which have to be disbelieved (OUT-set)
 to validate a justification.
- Disbelieved nodes may not be thrown away because the disbelief in a node may be used in a justification.

⇒ In a non-monotonic DDN, additionally the set of edges from nodes to justifications is partitioned into OUT-EDGE and IN-EDGE.

Labeling function on nodes:

- node labeled OUT is disbelieved
- node labeled IN is believed
- node labeled UNDET is undetermined
- A justification is
 - valid, if all nodes in the IN-set are believed and all nodes in the OUT-set are disbelieved nodes;
 - invalid, if one of the nodes in the IN-set is a disbelieved node or one of the nodes in the OUT-set is a believed node.

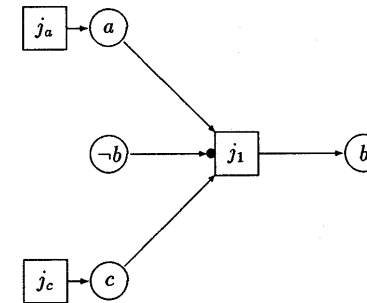
Example: TWEETY revisited. Let us assume the following propositions:

$a \equiv \text{Bird}(\text{TWEETY})$

$b \equiv \text{Fly}(\text{TWEETY})$

$c \equiv (\text{Bird}(x) : M\text{Fly}(x) / \text{Fly}(x))$

Under the assumption that $\{a, c\}$ is our default theory, a non-monotonic DDN including the default conclusion c , where the links in OUT-EDGE are marked by a dot, is given by



Problem:

- If the labeling of a justification changes from IN to OUT and the justification is the current support of a node, or
- if the labeling changes from OUT to IN and thus changes a node from OUT to IN

then

- all possibly affected labels are replaced by UNDET,
- the remaining partial labeling has to be turned into a complete, consistent, and well-founded labeling (relabeling procedure \Rightarrow NP-complete)

But there are reasonable classes of non-monotonic DDNs that are better behaved: If there are no

1. odd non-monotonic loops
2. even non-monotonic loops

we have the same complexity as in the monotonic case (the algorithm has to check for loops).

Assumption-Based Reason Maintenance

- Justification-based RMSs are inefficient in terms of space and time.
- It is impossible to compare the outcomes of different sets of assumptions.

De Kleer's ATMS (Assumption-based Truth Maintenance System):

- ▷ Label nodes with the sets of assumptions which were used in their derivation.
I.e., a node is neither believed or disbelieved, but believed relative to some context which is determined by sets of assumptions used in its derivation.

Basic ATMS Design

- Every datum the problem solver reasons about it assigned an ATMS node.
- The problem solver designates a subset of the nodes to be assumptions — nodes which are presumed to be true unless there is evidence to the contrary. The distinguished node \perp designates false.

The problem solver's search space is understood by a spatial metaphor — as being spanned by independent assumptions; the assumptions are the dimensions of the search space. A solution consists of the set of valid assumptions.

- Every relevant derivation made by the problem solver is recorded as a justification:

$$x_1, \dots, x_k \Rightarrow n$$

x_1, \dots, x_k : antecedent nodes

n : consequent node

- An environment is a set of assumptions. Each set of assumptions describes a subset of the search space, i.e. a context. The smaller this set is the bigger is the corresponding context.
- The ATMS does propositional reasoning over the nodes: Every node is a propositional symbol, and every justification is a Horn clause (a clause with at most one positive literal). An environment is a conjunction of propositional symbols.
- A node n is said to hold in an environment E if n can be propositionally derived from the union of E with the set of justifications. An environment is inconsistent ("nogood") if the node \perp holds in it. A nogood is minimal if it contains no others as a subset.
- The ATMS is incremental, receiving a constant stream of additional nodes, additional assumptions, additional justifications, and various queries concerning the environments in which nodes hold. It supports parallel operation on several context.

To facilitate these queries, it maintains for each node n a set of environments $\{E_1, \dots, E_k\}$, the label, having four properties:

1. Soundness: n holds in each E_i
2. Consistency: E_i is not nogood.
3. Completeness: Every environment E in which n holds is a superset of some E_i .
4. Minimality: No E_i is a proper subset of any other.

Given the label data structure the ATMS can efficiently answer the question whether n holds in E by checking whether E is a superset of some E_i . (The ATMS also maintains a database of all minimal nogoods.)

- The work of the problem solver is minimized, because information on already established contexts is kept and hence rederivations are avoided.

ATMS: Interface to the Problem Solver

Basic ATMS procedures:

- **ASSUME** : Generate an assumption for a problem solver datum.
- **NODE** : Generate a node for a problem solver datum.
- **JUSTIFY** : Add a justification to a node. This involves recomputation of its label and of the labels of all affected nodes.
- **LABEL** : Show the label of a node.
- **NOGOODS** : Compute the list of nogoods.
- **HOLDS?** : Ask whether a node belongs to a given context.

Extending the Basic ATMS

Motivation: Negation, default rules

Introduction of the concept of primitive disjunction to allow (non-Horn) justifications of the form

$$x_1, \dots, x_k \Rightarrow y_1, \dots, y_m$$

through a new operation $\text{CHOOSE}\{A_1, A_2, \dots\}$

Problems:

- Violation of monotonicity of derivation
- Complexity of the interpretation construction which yields the maximal consistent environments under consideration of disjunctions and nogoods.

Some Actual Problems

- Theoretical foundations of RMSs
Many RMSs still lack a clear implementation-independent specification of their behavior.
In general, clear and concise semantical theories of what belief revision and non-monotonic reasoning are all about are far from sight.
- Extensions to the ATMS
 - New approaches to allow an ATMS to handle more general clauses than Horn clause justifications (e.g. de Kleer's negated assumptions, CMS, RISC)
 - Focusing mechanisms for the ATMS to efficiently support problem solving in domains which are infinite and where the inference engine must retain tight control in the course of problem solving.
 - Parallel processing versions of the ATMS.