Chapter MK:V

V. Diagnoseansätze

- Diagnoseproblemstellung
- Diagnose mit Bayes
- Evidenztheorie von Dempster/Shafer
- Diagnose mit Dempster/Shafer
- Truth Maintenance
- Assumption-Based TMS
- Diagnosis Setting
- Diagnosis with the GDE
- Diagnosis with Reiter
- Grundlagen fallbasierten Schließens
- Fallbasierte Diagnose

Truth Maintenance

Non-Monotonicity

Situations where reasoning becomes non-monotonic:

- dynamic worlds, where “facts” can change
- observed input depends on time and place
- new input causes retractions, e. g., a hypothesis is discovered to be false
- exceptions become known

Foundations:

1. Deduction
   Collection: Logik, Part: Aussagenlogik
2. Non-Monotonicity
   Collection: Logik, Part: Ergänzungen

Remarks:

- The examples are typical for many diagnosis situations.
- If any of the mentioned situations occurs, one can simply restart the reasoning process from scratch in order to determine the valid deductions.
- A reason (or truth) maintenance system is some kind of bookkeeping mechanism that keeps track whether a fact is still inferable or not. The rationale is that the organizational overhead for bookkeeping is often smaller than an entire restart of the reasoning process.
- Truth maintenance system (TMS) in fact is a misnomer; rather say reason maintenance system (RMS).

Truth Maintenance

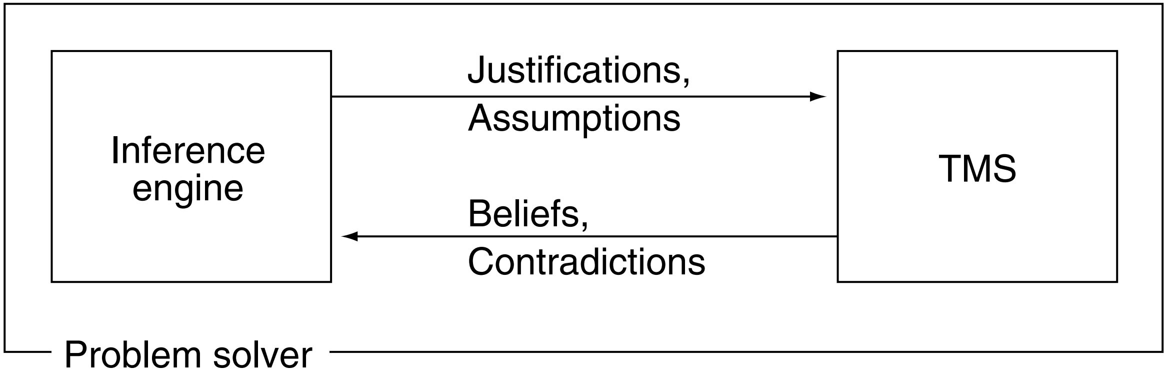
Operationalization

Idea: Identification and retraction of inferences that are no longer valid.

Well-known TMS concepts:

- dependency-directed backtracking
- justification-based truth maintenance (JTMS)
- assumption-based truth maintenance (ATMS)

Interplay between inference engine and TMS:

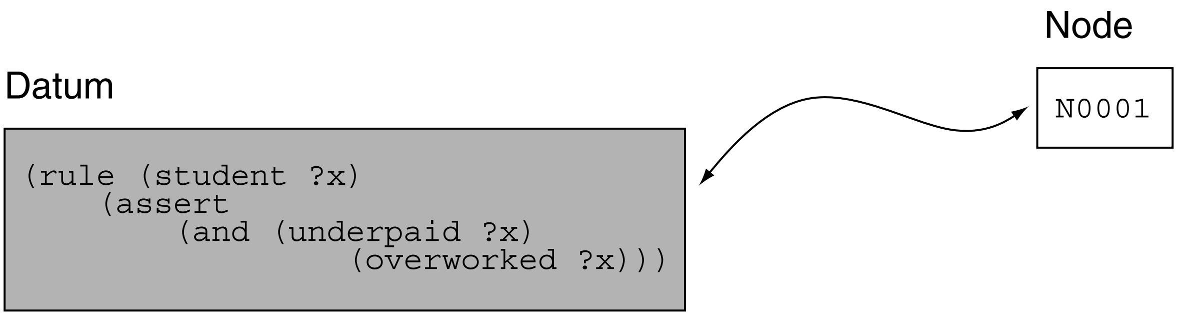


Truth Maintenance

Definition 12 (Datum, Node)

A datum is the smallest unit of information we are interested in. A node is the data structure of a TMS to represent a datum. There is a one-to-one correspondence between nodes and datums.

Illustration:



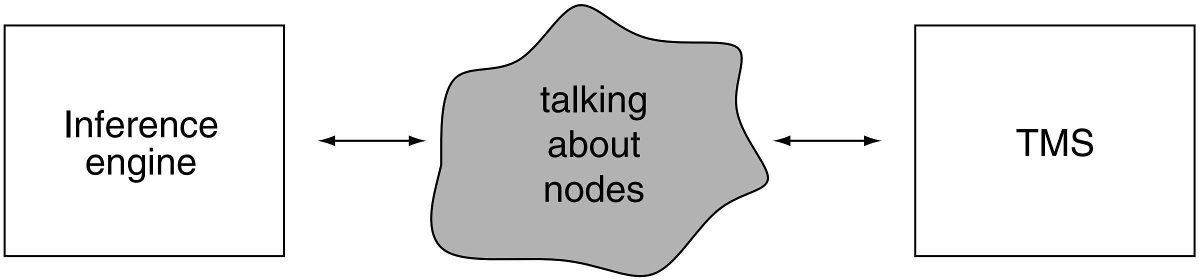
```
(rule (student ?x)
    (assert
        (and (underpaid ?x)
                 (overworked ?x)))
```

Examples for a datum:

- “The value of x is 20.”
- “Component y is defect.”
- “If x is true then y is defect.”
- “The roots of x^2 - 7 = 0 are 2.65 and -2.65.”

Truth Maintenance

Communication between the TMS and the inference engine is in terms of nodes:



Nodes are interpreted differently:

- Inference engine.
  Perform deductions that are grounded on the datum’s semantics.
- TMS.
  Perform deductions respecting a node’s validity. These deductions are grounded on logical implications.

Example for an inference engine deduction:

```
N0001:  IF student(X) THEN underpaid(X) AND overworked(x)
N0002:  student(robbie)
N0003:  underpaid(robbie) AND overworked(robbie)
```

Remarks:

- A datum is worth to be to be remembered / maintained / reasoned about / retracted / assumed. A datum can be an assertion, a fact, an inference rule, a procedures, a computation result, etc.
- The TMS cannot perform deductions that are grounded on the datum's semantics—but, the inference engine can inform the TMS on a deduction like $\texttt{N0001} \wedge \texttt{N0002} \vdash \texttt{N0003}$.

Truth Maintenance

The inference engine communicates the important (a subset of all) deductions to the TMS by means of justifications.

Definition 13 (Justification)

A justification is a condition on a node and consists of three parts:

1. Consequent. The node of the datum that has been inferred by the inference engine.
2. Antecedents list. Nodes of the datums that are used by the inference engine to infer the datum of the consequent.
3. Informant. A comment that explains the inference in more detail.

Example for an inference engine deduction:

```
N0001:  IF student(X) THEN underpaid(X) AND overworked(x)
N0002:  student(robbie)
N0003:  underpaid(robbie) AND overworked(robbie)
```

Justification communicated to the TMS:

```
⟨ N0003, MODUS-PONENS, {N0001, N0002} ⟩
```

Remarks:

- In the example, “`N0003`” is the consequent, “`MODUS-PONENS`” is the informant, and “`{N0001, N0002}`” are the antecedents.
- The example can be considered as a formulation in predicate logics.

Truth Maintenance

Definition 14 (TMS Node Types)

TMS nodes can be of the following types:

1. Premise.
   The *inference engine* has indicated that the associated datum is always true.

2. Contradiction.
   The *inference engine* has indicated that the associated datum is always false.

3. Assumption.
   The *inference engine* has indicated that the associated datum is true unless stated otherwise.

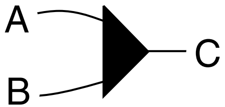
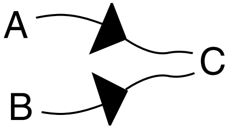
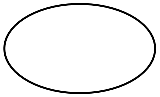
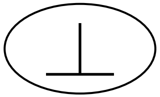
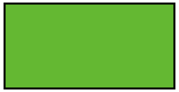
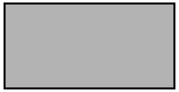
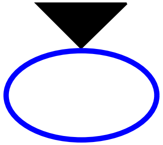
4. Justified.
   Based on premise nodes, assumption nodes, and justifications, the node can be logically inferred by the *TMS*.

Remarks:

- With each node a label is associated; it encodes the current truth value (belief) of its node.
- Assumption nodes are used to represent datums which the inference engine has chosen to believe at the moment, but which it may want to retract later.
- The associated datum of a justified node is true.
- The TMS informs the inference engine if a node is justified or not, and if a contradiction can be inferred.
- Depending on the associated node's type, the colloquial semantics of the datum "`student(robbie)`" is as follows:
  - Robbie is a student for ever (premise).
  - Robbie is not student (contradiction).
  - Let's say, Robbie is a student (assumption).
  - The TMS can infer that Robbie is a student (justified).

Truth Maintenance

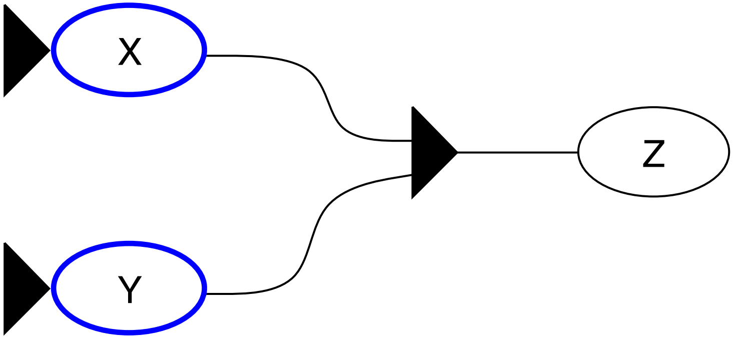
Graphical notation, adopted from [Forbus/deKleer 1993]:

Symbol	Semantics
	justification $A \wedge B \rightarrow C$
	two justifications $A \rightarrow C,\ B \rightarrow C$
	justified node
	contradiction node
	assumption node (enabled)
	assumption node (retracted)
	premise node

Truth Maintenance

How Justifications Help

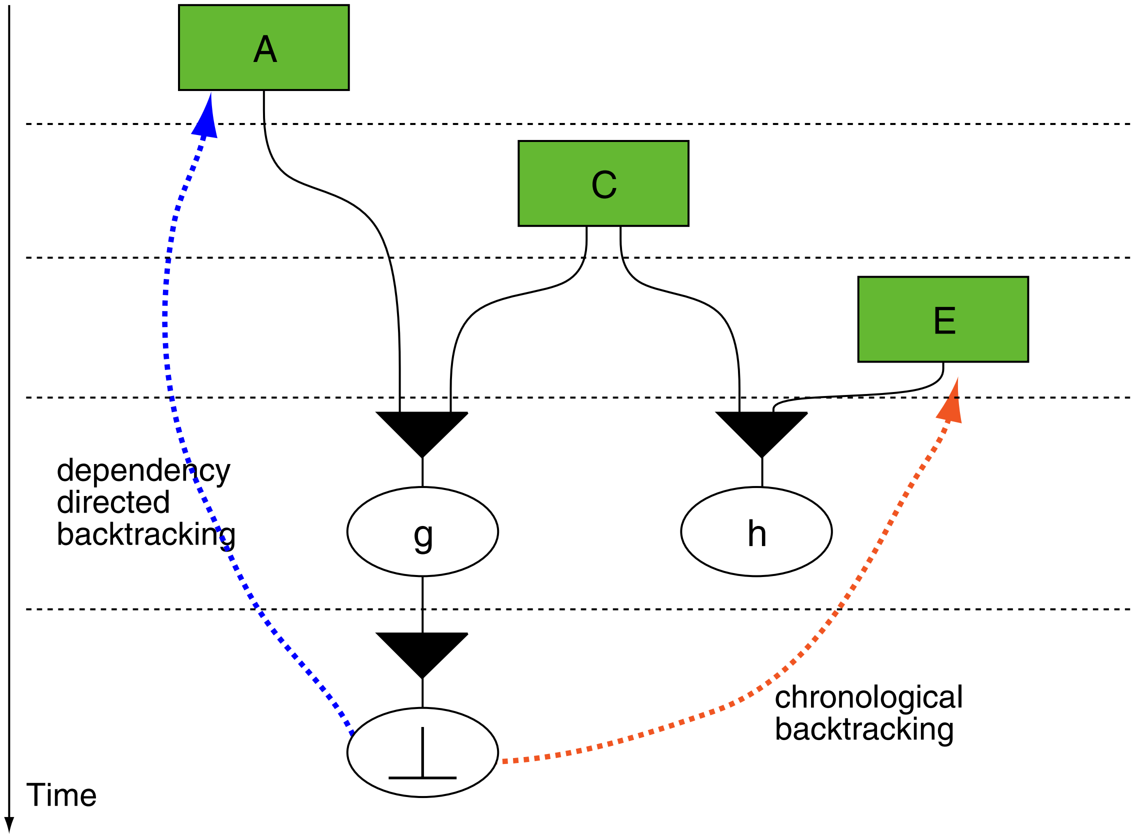
Identifying responsibilities for conclusions: Why does Z hold?



Truth Maintenance

How Justifications Help (continued)

Guiding backtracking: Which assumption shall be retracted?



Truth Maintenance

Formalization

Specification of TMS concepts in propositional logics:

- Every TMS node can be interpreted as a propositional symbol.
- Every justification can be interpreted as a propositional definite Horn clause. If the nodes $x_1, \ldots, x_m$ justify node n, this is represented as:

$$\neg x_1 \vee \ldots \vee \neg x_m \vee n \quad \approx \quad (x_1 \wedge \ldots \wedge x_m) \to n$$

- A premise node n corresponds to a unit clause n.
- A contradiction node can be specified by the negative unit clause $\neg n$.

Truth Maintenance

Label

The truth value (belief) of a node is stored in its label; it can be encoded in different ways.

Concepts of a JTMS:

- A node can be labeled either as `:IN` or as `:OUT` and indicates whether or not it can be logically inferred from premise nodes, assumption nodes, and justifications.
- The JTMS answers queries respecting a node's truth value by simply returning its label.

Problems when using a JTMS:

- Assumption changes require relabeling.
- Cyclic justifications must be detected and treated specifically.

Remarks:

- If the rate of assumption changes is much larger then the number of queries about node labels, much time is wasted in relabeling.
- The JTMS is unsuitable as truth maintenance concept if the problem solving task requires frequent context switches or the parallel analysis of several contexts.

Truth Maintenance

Label (continued)

Concepts of an ATMS:

- The label of a node n contains sets of assumption nodes. Using the given premises and justifications, n can be logically inferred from each set of assumption nodes.
- The ATMS answers queries respecting a node's truth value by checking the node's label. Example: Can node n be inferred from the assumptions $\{A, B, C\}$?
- If a contradiction node can be inferred from a set of assumptions, this set is removed from all labels.

Problem when using an ATMS:
The size of a node's label, i. e., the number of sets of assumptions where this node holds in, can grow exponentially in the number of assumption nodes.

Remarks:

- The inference engine should introduce as few assumptions as possible.
- The ATMS ensures that no node follows from a set of assumptions if a contradiction node also follows from that set. This implies that no special contradiction handling is necessary for the ATMS.

Truth Maintenance

JTMS versus ATMS

JTMS.

A context switch means:

1. Relabel assumptions respective desired `:IN` or `:OUT` value.
2. Retract justifications that are no longer valid.
3. Propagate justifications that are valid now.

➜ JTMS incurs the cost of context switching when the actual context switches are made.

ATMS.

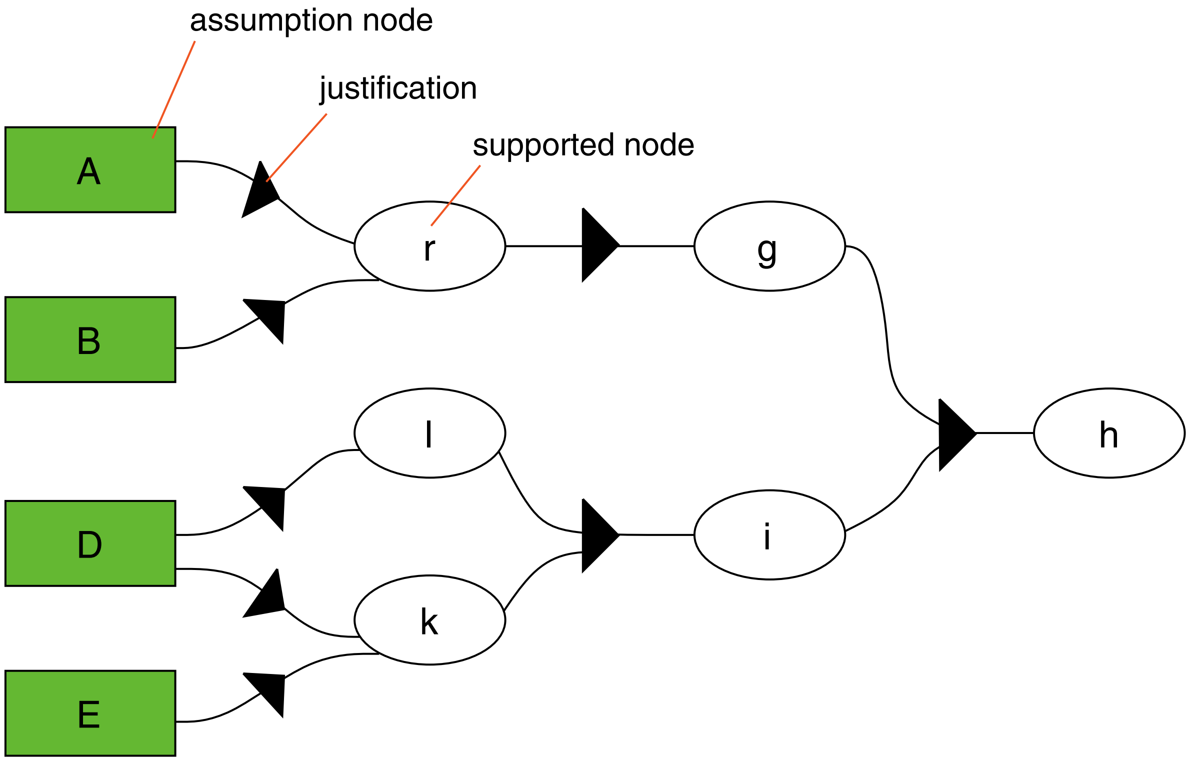
Context switches are free since all reasonable contexts are constructed already.

A “reasonable context” is a context from which something can be inferred.

➜ ATMS incurs the cost of context switching up front.

Assumption-Based TMS (ATMS)

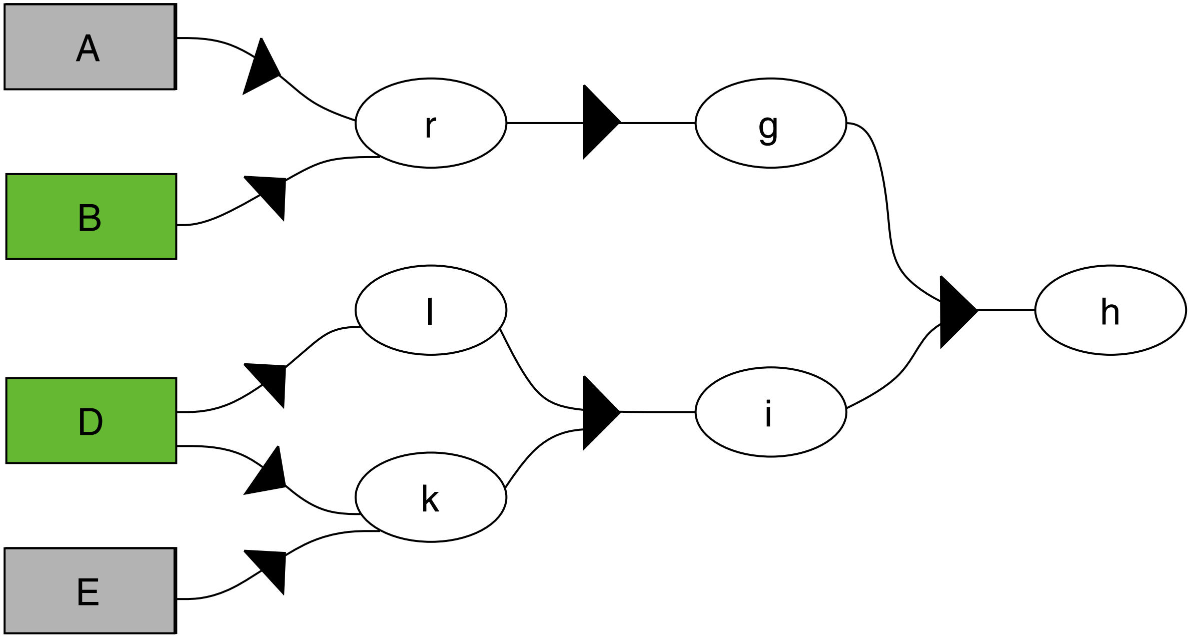
Example



- Assumptions made: $\{A, B, D, E\}$
- Result: h follows from $\{A, B, D, E\}$

Assumption-Based TMS

Example (continued):



- Retract: A, E
- Result: h follows from $\{B, D\}$

In particular, h follows from:

$\{A, D\}, \ \{A, B, D\}, \ \{A, D, E\}, \ \{A, B, D, E\}, \ \{B, D\}, \ \{B, D, E\}$

Assumption-Based TMS

Terminology

- Environment.

  A set of assumption nodes, short: assumptions.

- Holds / is supported.

  A node holds in / is supported by an environment $\mathbf{E}$, if there exists a set of justifications such that the node can be logically inferred from $\mathbf{E}$.

- Label (of a node).

  A label of a node comprises environments where the node holds in.

- Nogood set.

  An environment in which some contradiction node holds.

- Consistent.

  A consistent environment is one which is not a nogood set.

- Context (of an environment).

  A context of an environment is the set of nodes that hold in the environment.

Remarks:

- The node h holds in six environments.
- The context of the assumptions $\{A, B\}$ is $\{r, g\}$.

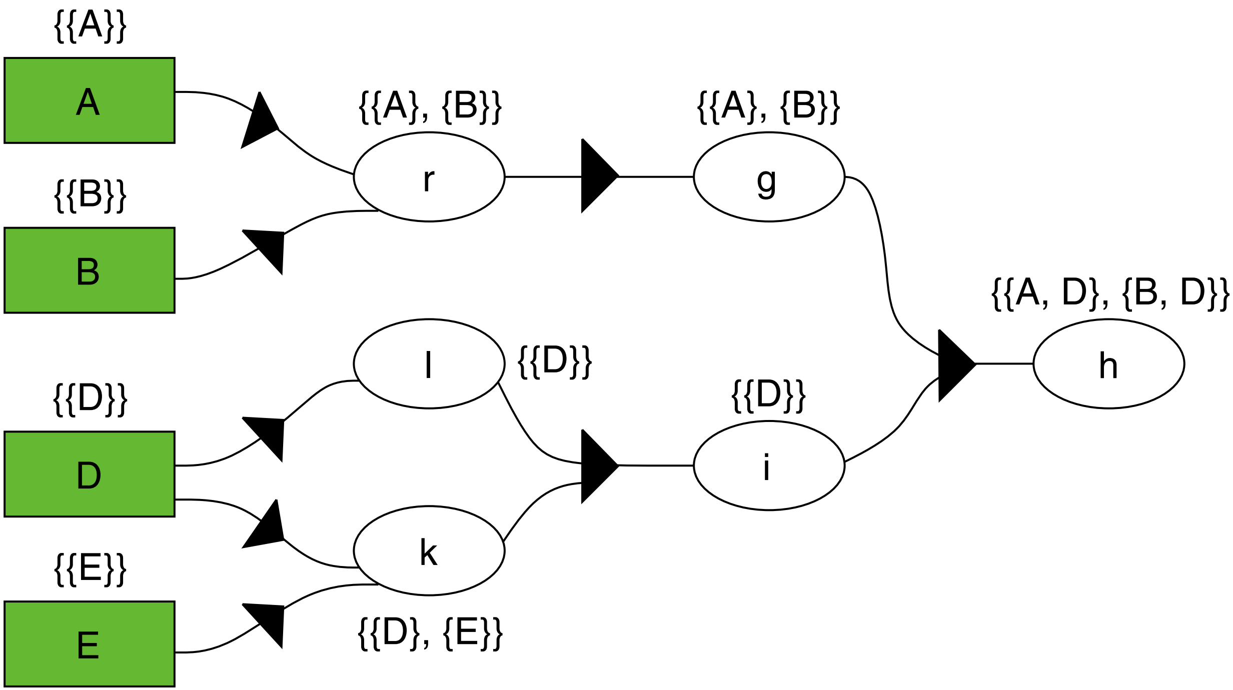
Assumption-Based TMS

ATMS Labels

Objective: Determine whether a node n holds in some environment.

Naive solution: Record all environments in which n holds.

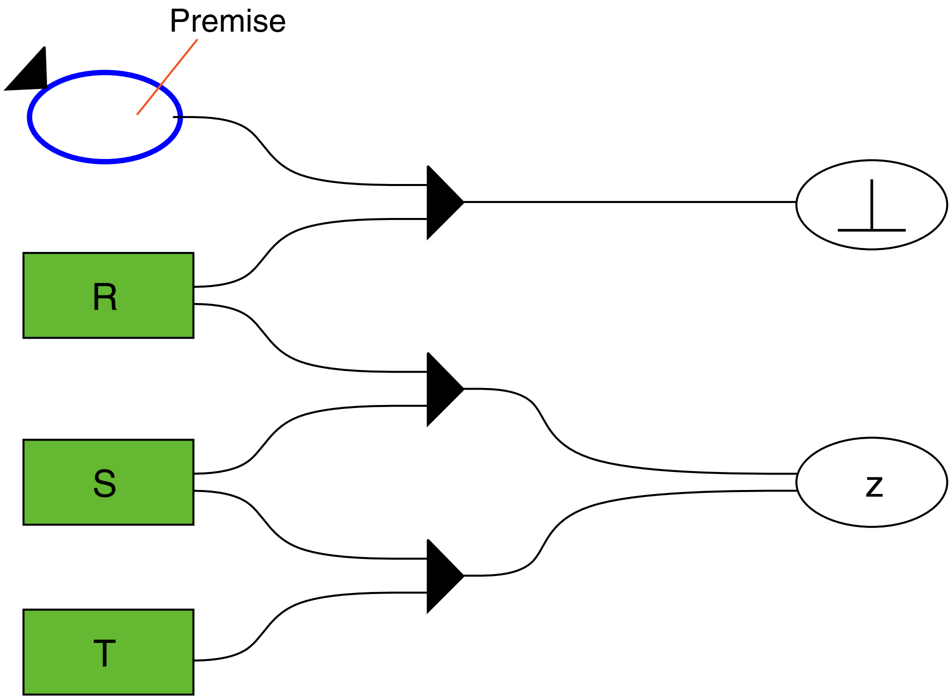
Better: Exploit monotonicity: If n follows from some environment $\mathbf{E}$, it will follow from any superset of $\mathbf{E}$ as well. Discard nogood sets from a node's label.



Assumption-Based TMS

ATMS Labels (continued)

Example:



- The label of z is $\langle z, \{\{S, T\}\}\rangle$.
- $\{R, S\}$ seems to support node z, but is not included in the label because it is a nogood set. More precisely:

  $\{R, S\}$ contains an environment, $\{R\}$, which is a nogood set.

Assumption-Based TMS

ATMS Labels (continued)

Following labels play a special role:

(a) Label with no environment: $\langle n, \{\}\rangle$

- The dependency network of justifications contains no pathway from the set of assumptions and premises to the node n, or
- all potential label environments are nogood sets.

(b) Label with the empty environment: $\langle n, \{\{\}\}\rangle$

- The node n holds in every environment.

Remarks:

- If a node n has a label with no environment then n has no support; it does not mean that n is a contradiction.
- All premises have a label consisting of the empty environment.
- Non-premise nodes can hold in empty environments if they ultimately depend only on premises.

Assumption-Based TMS

Summary of ATMS concepts in propositional logics:

- The ATMS nodes define a set of propositional symbols Σ.
- A subset $\mathbf{A} \subset \Sigma$ of symbols are marked as assumptions.
- An ATMS justification is encoded as a definite Horn clause.
- If n is a premise node, it is represented as unit clause n.
- If n is a contradiction node, it is represented as the negative unit clause $\neg n$.
- An environment $\mathbf{E}$ is a subset of assumptions, $\mathbf{E} \subset \mathbf{A}$.

Assumption-Based TMS

Definition 15 (Hold, Nogood Set, Miminality)

Let α be the logical conjunction of premises and justifications, and let α be consistent. Moreover, let β be the logical conjunction of the assumptions of an environment $\mathbf{E}$.

1. A node n is said to hold in the environment $\mathbf{E}$, if $\alpha \wedge \beta \models n$.
2. If $\alpha \wedge \beta \models n$ and if n is a contradiction node, then the environment $\mathbf{E}$ is a nogood set.
3. A nogood set is minimum if it contains no other nogood set as a subset.

Assumption-Based TMS

Definition 15 (Hold, Nogood Set, Miminality)

Let α be the logical conjunction of premises and justifications, and let α be consistent. Moreover, let β be the logical conjunction of the assumptions of an environment $\mathbf{E}$.

1. A node n is said to hold in the environment $\mathbf{E}$, if $\alpha \wedge \beta \models n$.
2. If $\alpha \wedge \beta \models n$ and if n is a contradiction node, then the environment $\mathbf{E}$ is a nogood set.
3. A nogood set is minimum if it contains no other nogood set as a subset.

ATMS recording service:
The ATMS receives a stream of nodes, assumptions, and justifications.

ATMS information service:
The ATMS answers queries concerning environments and nodes that may hold in these environments.

Assumption-Based TMS

Definition 16 (ATMS Label Properties)

To facilitate answering queries, the ATMS maintains for a label $\{\mathbf{E}_1, \ldots, \mathbf{E}_k\}$ of some node n the following four properties:

1. Soundness.
   n holds in each $\mathbf{E}_i$.

2. Consistency.
   No contradiction node can be derived from some $\alpha \wedge \beta_i$. α is the conjunction of recorded premises and justifications, and β_i is be the conjunction of the assumptions in $\mathbf{E}_i$.

3. Completeness.
   Every consistent environment E in which n holds is a superset of some $\mathbf{E}_i$.

4. Minimality.
   No $\mathbf{E}_i$ is a proper subset of any other.

Assumption-Based TMS

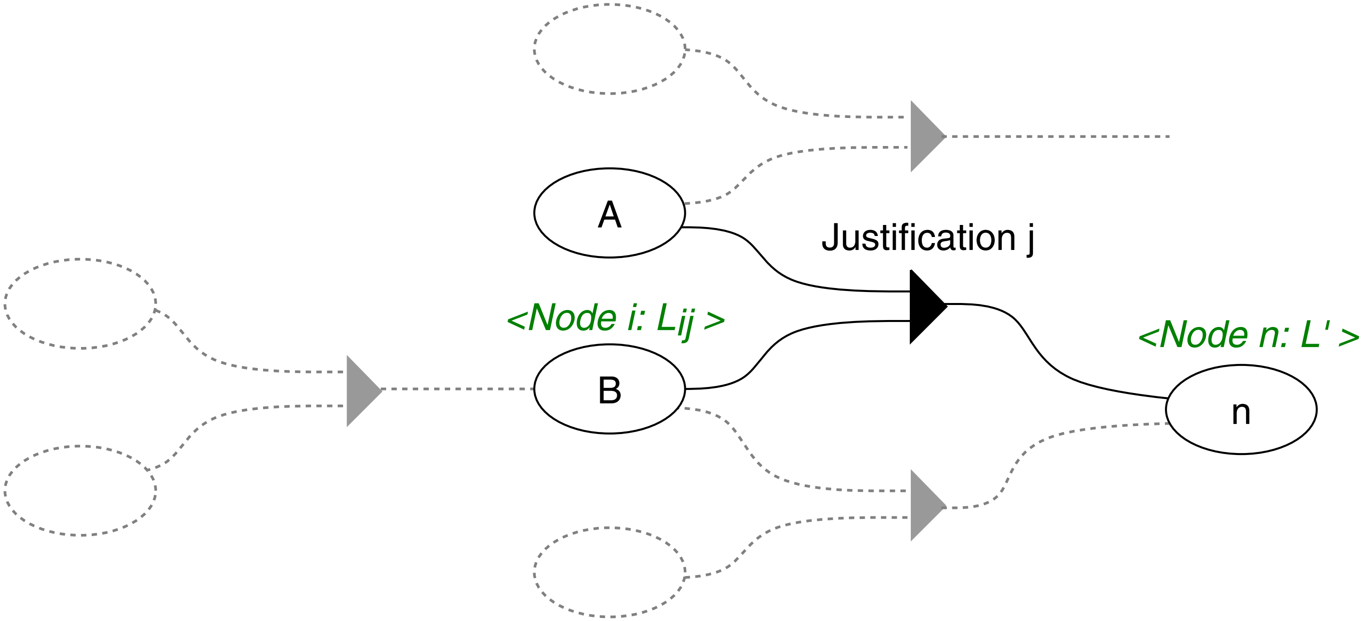
Label Update

Concepts:

- Label updating happens incremental.
- Labels are made locally correct and label changes are propagated until labels become globally correct.
- Assumptions are created with labels containing the single environment containing themselves.
- All other nodes are created with empty labels.

Assumption-Based TMS

Label Update (continued)



Let L_{i_j} be the label of the ith node of the jth justification for some node n. Label update algorithm for node n:

1. Compute a tentative label $L' = \{\bigcup_i e_i \mid e_i \in L_{i_j}\}$
2. Remove from L' all nogood sets, and all environments subsumed by other environments in L'.

Assumption-Based TMS

Label Update (continued)

Example:

- $\langle e, \{\{A, B\}, \{C\}\}\rangle$
- $\langle f, \{\{A\}, \{D\}\}\rangle$
- $\langle \perp, \{\{C, D\}\}\rangle$

New justification j from inference engine: $e \wedge f \rightarrow g$

Assumption-Based TMS

Label Update (continued)

Example:

- ❑ $\langle e, \{\{A, B\}, \{C\}\}\rangle$
- ❑ $\langle f, \{\{A\}, \{D\}\}\rangle$
- ❑ $\langle \bot, \{\{C, D\}\}\rangle$

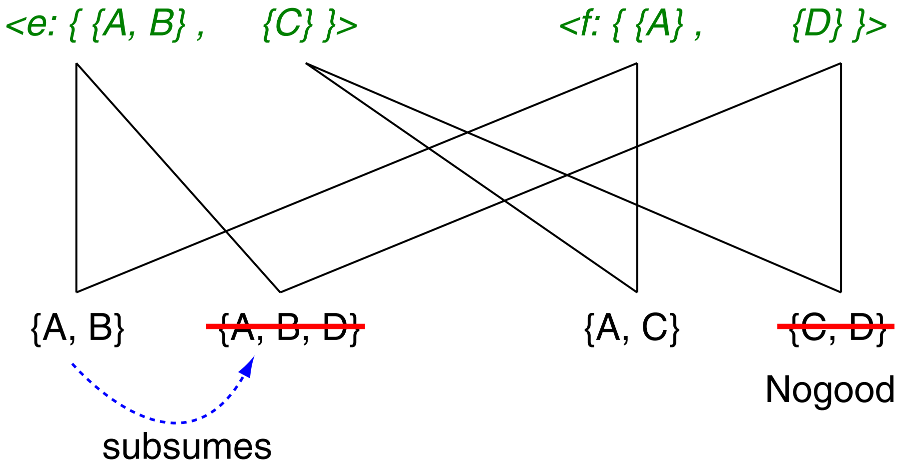
New justification j from inference engine: $e \wedge f \rightarrow g$

Computation of a new label for node g:

$L_{1_j} = \{\{A, B\}, \{C\}\}$, $L_{2_j} = \{\{A\}, \{D\}\}$

1. $L' = \{\{A, B\}, \{A, B, D\}, \{A, C\}, \{C, D\}\}$
2. Remove $\{C, D\}$ and $\{A, B, D\}$

Correct label for g: $\{\{A, B\}, \{A, C\}\}$



Assumption-Based TMS

Label Update (continued)

Global label update algorithm:

1. To update node n, compute its new label L' as described before.
2. If the label is not changed, return.
3. If n is a contradiction node:
   (a) Mark all environments of L' as nogood sets.
   (b) Remove all new nogood sets from every node label.
4. If n is not a contradiction node, then recursively update all the consequences of n.

Remarks:

- This algorithm is inefficient (cf. Step 1). A more efficient version propagates merely incremental changes of node labels.