FOUNDATIONAL CONCEPTS: TECHNICAL BRIEF

Truth Maintenance Systems

A truth maintenance system tracks the reasons for each belief in a reasoning system.

The system stores each belief with its justifications.

The system marks a belief IN after all premises hold.

The system marks a belief OUT after one premise fails.

The system changes dependent beliefs in a cascade after a premise change.

Modern agent memory stores facts without justification links.

This brief describes the old system and proposes a new project.

Start Here: The Short Version

This method helps a computer track why it believes each fact. Each fact keeps links to the facts and sources under it. A fact can rest on a file, a record, or another fact. The system checks those links as work goes on. A source changes or a key fact fails. The system then removes the beliefs that no longer have support. Other beliefs stay in place when support still holds.

A simple example

> Think of a simple case in a work file. A note says a server runs one program version. A second note uses that fact to plan a fix. The source file then changes to a new version. The system marks the first note as out of date. The system clears the plan that rests on that note.

What it can do

- It shows the reason chain behind an answer. A person can follow each link back to the source.
- It removes stale beliefs in a clean chain after a change. One change can clear many linked beliefs at once.
- It records each removal in a log. A person can audit the change later and see the cause.

Where people can use it

- AI agent memory must forget facts from changed source files. The system clears the old facts when a file changes.
- Compliance work needs a source for each claim. The system keeps the source link with each claim.
- Design and diagnosis tools link many parts in one plan. One part change can affect many linked parts. The system tracks each effect of that change.

Problems it can help solve

- AI systems store false facts and use the false facts again later. Bad facts spread into new answers over time.
- Source data changes, but old answers stay in memory. Teams then work from facts that no longer hold.
- Teams cannot explain why the system gave an answer. Work slows down when no one can check the reason chain.

The rest of this brief gives the history, the method, the 2026 gap, and a first project plan.

1: Identification

Full name	Truth Maintenance System (TMS)
First form	Justification-Based TMS (JTMS), Jon Doyle, 1979
Second form	Assumption-Based TMS (ATMS), Johan de Kleer, 1986
Core data	Beliefs, justifications, premises, labels IN and OUT
Core actions	Label, trace, retract, record a nogood
Main users in the 1980s	Expert systems, diagnosis systems, design tools

2: Purpose

Reasoning systems derive new beliefs from old beliefs and from source facts.

Source facts change during long work.

A derived belief loses its support after a source fact changes.

The TMS keeps the belief set consistent after each change.

The TMS also explains each belief through its justification chain.

Diagnosis tools trace faults through this chain during a fault search.

3: Operation

A nogood names a premise set with a known contradiction.

Numbered steps: one belief

1. The problem solver sends a new belief to the TMS.
2. The problem solver sends the justification for the belief at the same time.
3. The justification lists the premises that support the belief.
4. The TMS stores the belief, the justification, and the source links.
5. The TMS sets the label IN for a belief with full support.
6. The TMS sets the label OUT for a belief with one failed premise.
7. A contradiction starts dependency-directed backtracking in the TMS.
8. The TMS traces the contradiction to the premises under the contradiction.
9. The TMS retracts one premise set and records the set as a nogood.
10. The TMS changes labels along the dependent paths only.
11. Other branches keep their labels and their work.

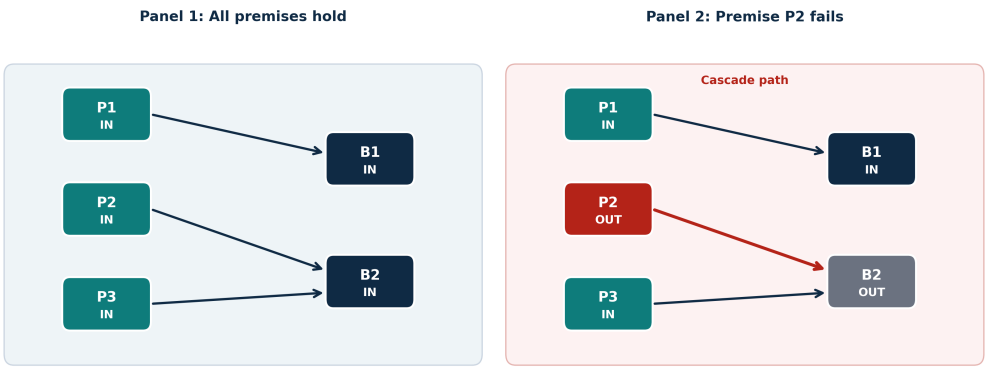

Figure 1: A justification network. Premise P2 fails, and belief B2 loses support.

> The JTMS holds one active context at a time.
>
> The label shows the state of the belief in that context.

3: Operation, The ATMS Form

The ATMS tracks many contexts in parallel.

Each context holds one set of assumptions.

The ATMS calls each set an environment.

Each belief carries a label with the environments for the belief.

The same belief holds in one environment and fails in another environment.

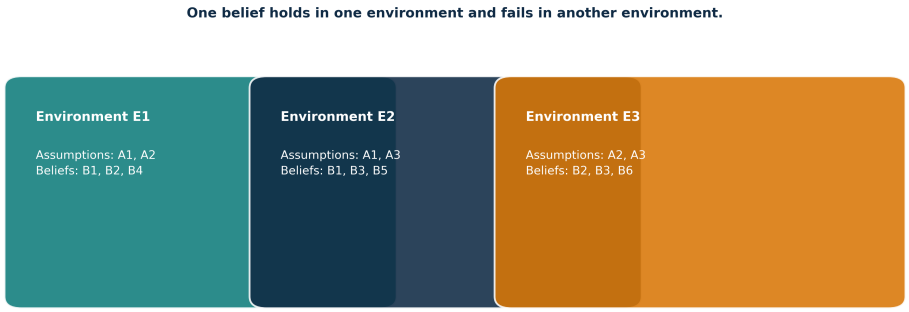

Figure 2: ATMS environments. Each environment lists the beliefs that hold under its assumptions.

4: History

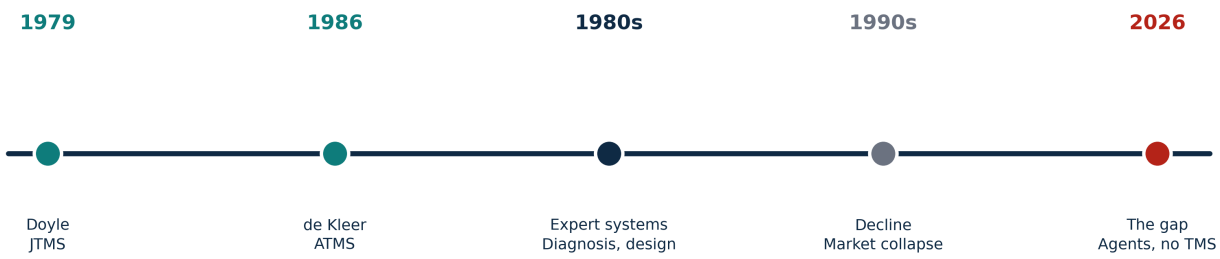

Figure 3: Timeline. The system peaks with expert systems and declines in the 1990s.

Expert systems used TMS technology in the 1980s.

Diagnosis systems tracked fault hypotheses with TMS labels.

Design tools tracked design choices and their reasons.

The expert-system market collapsed in the late 1980s and early 1990s.

The collapse removed the main buyers for TMS software.

The bookkeeping cost grew with network size.

Large networks needed large label sets in the ATMS form.

Probabilistic methods took the uncertainty field during the same period.

Research teams chose probability tools for new projects.

5: The 2026 Gap

Language-model agents store facts in memory stores and in chat histories.

Retrieval-augmented generation systems store text chunks in vector stores.

These stores keep source links and similarity scores.

These stores keep no justification links between facts.

No mainstream product removes derived beliefs in an automatic cascade after a source change.

No mainstream product removes derived beliefs in an automatic cascade after a premise failure.

> A false stored fact stays in memory for months.
>
> Later answers use the false fact as support.
>
> The system presents the answer without a warning.

Small research projects test belief layers in 2026.

These projects still lack a standard JTMS engine or ATMS engine in the main frameworks.

Agent frameworks in 2026 include no standard JTMS engine and no standard ATMS engine.

6: Concrete Use Cases

Each case shows one belief, the premises under the belief, one change, and the cascade after the change.

Stale agent memory

An agent holds the belief that reports go to one named contact at a client firm. The belief rests on one customer record in the CRM system. The record changes to a new contact after the first contact leaves the firm. The belief drops, and the scheduled report task flags for review instead of sending the report to the old contact.

Runbook safety

A runbook holds the belief that a host is safe for a patch and reboot step. The belief rests on a completed backup record and on an empty production workload list. The backup record corrects to a failed state. The safety belief drops, the auto approval for the step revokes, and the ticket escalates to a human.

Fraud and compliance screening

A screening file holds the belief that a provider is low risk. The belief rests on an active enrollment record, an address match, and a clean exclusion check. The enrollment record lapses. The low risk belief and the no review belief drop, the file returns to the review queue, and the audit log shows the changed premise.

Research integrity

A published report holds one figure as a stated claim. The claim rests on one cited source paper. The source paper issues a correction. The layer lists every claim and every document that rests on the source, and editors fix the full set in one pass.

Quotes and eligibility

A sales system holds the belief that a quote is valid for send. The belief rests on a discount approval, a price list version, and a contract pricing status. The discount approval expires. The quote belief drops, and the system blocks the stale quote before a send.

A parallel form keeps several assumption sets apart at the same time. Plans under different supplier timing assumptions stay in separate worlds, and each output carries its own assumption set.

7: Concrete Examples

Section 6 states each case in short form. This section works six cases in full. Each case names the belief, the premises, the change, and the cascade in order. Each case ends with the limit of the layer in that setting.

A diagnosis and its treatment cascade

A care team treats a patient for bacterial pneumonia. The diagnosis is a belief in the chart system. Four premises support the diagnosis.

- A chest image read reports consolidation in the right lower lobe.
- A lab report gives a high white cell count.
- The history record lists fever for three days.
- A sputum culture grows a named bacterium.

The imaging team retracts the image read after a second review. The second review finds atelectasis and no consolidation. The first premise fails, and the diagnosis loses its strongest support. The label engine marks the diagnosis OUT. The antibiotic order rests on the diagnosis alone. The engine marks the antibiotic order OUT and flags the order for clinician review. The treatment plan cites the diagnosis in three sections. Each section loses support and retracts in the same cascade. The follow-up image order rests on the treatment plan. The engine marks that order OUT. The fever care order rests on the temperature record alone. That premise still holds, and the order stays IN. The justification chain answers the audit question for the antibiotic order. A reviewer follows the links to the premise values in force on the order date. The layer tracks support and makes no diagnosis. Clinicians choose the replacement diagnosis and the new treatment.

An intelligence profile after a source correction

An intelligence profile assesses the health and the support base of a foreign leader. The profile holds each judgment as a belief with named sources. Three source reports support the judgment that a named faction backs the leader. Each report carries a reliability note from the handling team.

- Source A reports faction meetings with the leader.
- Source B reports faction funding for the leader.
- Source C reports public statements of support.

The agency discredits Source B after a fabrication check. The funding report retracts, and its premise fails. The funding judgment rests on the Source B report alone. The engine marks the funding judgment OUT. The succession assessment cites the funding judgment as one premise. The assessment loses that premise and flags for re-derivation. The health judgment rests on reports from Source A and Source C. Those premises still hold, and the health judgment stays IN. An analyst requests the dependency list for Source B. The layer returns every judgment with a link to that source in one list. The layer marks reach and support only. Analysts re-derive each flagged judgment by hand.

7: Concrete Examples, Plans and Memory

A battlefield plan after an assumption fails

A brigade staff holds one course of action as the active plan. The plan is a conclusion in an assumption-based network. Four assumptions support the plan.

- A route survey rates the river bridge for heavy vehicles.
- A reconnaissance report places the enemy reserve unit at one named town.
- A weather report gives a clear window for two days.
- A readiness report lists two battalions at full strength.

A new reconnaissance report corrects the enemy unit position. The unit sits on the planned route, and the second assumption fails. The plan label loses the environment with the failed assumption. The engine marks the plan unsupported in the current picture. Three orders cite the plan as their premise. Each order flags for review before execution. A second plan rests on a different assumption set. That set still holds, and the second plan stays active. Command receives the list of affected orders in one pass. The layer lists exposure and gives no tactical advice. Commanders choose the replacement course of action.

Agent memory after a source change

An operations agent stores facts from a project plan file. Each fact carries a link to the file section as its premise. The file states a deployment window on Friday at 18:00. The agent derives a freeze start on Thursday from the window. The agent derives a notice deadline on Wednesday from the freeze start. A calendar hold and a team notice rest on the deadline. The plan owner edits the file and moves the window to Saturday. The stored window fact loses support and drops. The freeze start and the notice deadline drop in the same cascade. The calendar hold and the team notice flag for rewrite. Facts from the staffing roster file keep support and stay IN. The audit log records each removal with the file edit as the cause. The layer clears derived facts and writes no replacements. The agent rebuilds its schedule from the new file version.

7: Concrete Examples, Claims and Design

Compliance claims after an enrollment lapse

A compliance file holds eligibility claims for one care provider. Each claim rests on an enrollment record and an exclusion check. The enrollment record covers a stated period. The claims support service authorizations and payment requests. The payer reports a lapse in the enrollment record. The lapse covers the last two months of the period. The engine marks each claim inside the lapse window OUT. Authorizations that cite those claims flag for review. Payment requests in the same window flag for hold. Claims before the lapse keep full support and stay IN. The audit log lists each flagged item with the lapse as the cause. The layer flags exposure and decides no eligibility question. Compliance staff resolve each flagged claim with the payer.

Design claims after a component spec change

An enclosure design holds a thermal claim for a handheld device. The claim states a surface limit under full load. Three premises support the thermal claim.

- The processor data sheet gives a peak draw of five watts.
- A board layout claim fixes the heat path to the rear shell.
- A material claim gives the shell a stated heat rating.

The vendor revises the data sheet and raises the peak draw to seven watts. The first premise fails, and the thermal claim drops OUT. The fan selection claim cites the thermal claim. The battery life estimate cites the peak draw and the thermal claim. Both claims lose support and flag for rework in order. The certification claim cites the thermal claim as evidence. The certification claim flags for hold before submission. Claims for other components keep their premises and stay IN. The layer propagates the change and makes no design choice. Engineers set the new heat path and the new fan.

The six cases share one pattern. A premise falls, the layer traces the links, and dependent claims retract in order. Claims with support from other premises stay in place in each case.

8: Alignment and Safety

Alignment work needs three records for every agent claim. Each claim needs its reasons, its assumption set, and a retraction path. A truth maintenance layer holds all three records in one graph.

Assumption hygiene

An agent in a test run holds the assumption that all targets are simulated. Each conclusion from the test run carries that assumption as a label. A real action request outside the test context finds no support in the graph. The action stops at the gate instead of running on a test assumption.

Instruction change and injection

A plan rests on one instruction from a principal and one fact from a tool feed. The principal replaces the instruction with a new instruction. Every plan step under the old instruction retracts in one cascade. A separate check flags any high-stakes step with support from untrusted page text alone.

Living safety cases

A release claim rests on an evaluation result, a red team round, and a monitoring feed. The evaluation result falls after a benchmark leak report. Every safety claim under the result flips to unsupported in the graph. The release gate reads the graph and blocks the release.

Compromised source reach

A tool feed enters the graph as a premise source for many beliefs. A review finds the feed compromised at a stated date. The graph lists every decision and belief with support from the feed. Oversight teams review the full list instead of sample checks.

False consensus and self support

Many agents state the same claim in a shared task. The graph shows one unverified output as the only source under all copies. Reviewers count the claim as one source, not many. A second check flags any belief with a support chain that loops through agent outputs alone.

Limits

The layer records stated reasons for a claim. Stated reasons differ from the real internal causes of a model output. A model with fluent text output builds clean graphs for weak decisions. Teams apply the layer at decision level, not at token level, and humans define the valid justification types.

9: Project Proposal

Goal. The desired end state is one justification layer under agent memory in daily use. Every stored belief carries its premises and its sources. Every source change triggers a check of all beliefs that rest on that source.

Proof target. The project proves one claim with an exact retraction test. After a source change, the layer finds and removes or recomputes 100 percent of the beliefs that rest on the changed source. Agent answers show fewer stale fact errors than answers from plain memory storage.

Product at the end. The product is the Justification Layer, version 1. The layer contains a graph belief store, a justification writer, a label engine, a change monitor, and an audit log. Agent memory systems place the layer under the memory store.

The project adds a justification layer to agent memory.

Each stored belief carries premise links and source identifiers.

A source monitor detects a change in a source document or a database value.

The layer recomputes labels after the change.

The layer removes the beliefs with failed premises in a cascade.

The layer writes each removal to an audit log with the cause.

Components

Belief store	The belief store holds beliefs, premises, and justification links in a graph.
Justification writer	The justification writer runs at storage time and records premise links and source identifiers.
Label engine	The label engine recomputes IN and OUT labels after each change.
Change monitor	The change monitor watches source documents and database values for changes.
Audit log	The audit log records each removal with its cause and its time.

9: Project Proposal, Flow and Test

Numbered steps: one source change

1. An agent writes a belief to the belief store.
2. The justification writer attaches premise links and a source identifier to the belief.
3. The label engine sets the first label for the belief.
4. The change monitor detects a change in a source document or a database value.
5. The label engine recomputes labels for all dependent beliefs.
6. The layer removes each belief with a failed premise in a cascade.
7. The audit log records each removal and the failed premise.

Build order

1. Build the label engine and the nogood store.
2. Build the graph belief store.
3. Build the justification writer for two source types.
4. Build the change monitor for files and database rows.
5. Build the audit log and a label viewer.
6. Test the layer on incident records and on project documents.

The test

The test changes one source fact in a test set.

The layer removes exactly the dependent beliefs in the test.

The layer keeps all other beliefs in the test.

The audit log names the failed premise for each removal.

10: Risks and Limits

Bad justifications produce bad cascades.

A language model writes weak premise links for hard claims.

Human review protects high-value beliefs from weak links.

ATMS label sets grow fast in large networks.

The prototype limits each environment set to a small size.

The TMS tracks support and gives no truth score.

A belief with full support still rests on false premises in some cases.

The layer fits structured claims first.

Free text claims need a writer with strict extraction rules.

Verdict

The project fills a clear gap in agent memory.

The core engine uses a small and proven algorithm.

The first prototype needs one data set and one change test.

Sources: Doyle, J. (1979). A Truth Maintenance System. Artificial Intelligence 12(3). de Kleer, J. (1986). An Assumption-Based TMS. Artificial Intelligence 28. Foundational Concepts series research notes, 2026-10-23.