A SCIENTIFIC AND INTELLECTUAL HISTORY

History of Diagnostics, Observability, and Debugging in Formal Sciences

Mathematical arguments and machine checked evidence

— Mathematical artifacts and their analysis —

Concept and direction by Dmitry Vostokov, DumpAnalysis.org

Developed with AI assistance (GPT-6 Astra Max and Fable 5.1 Extra) and checked against the sources listed in this article.

Version 5 | September 2026

Historical synthesis with 82 references, chronology, glossary, and evidence taxonomy

Abstract

Formal sciences provide methods for finding errors in arguments, calculations, models, and specifications. They also provide results about what cannot be inferred or decided under particular assumptions. In this history, we consider both developments. A counterexample can refute a conjecture. A proof can expose an unnecessary assumption. A residual can reveal a limitation of a statistical model. An unobservable state can explain why additional processing of the same outputs does not resolve a diagnostic question.

We follow these activities through mathematics, logic, probability, statistics, numerical analysis, information theory, control theory, optimization, and the foundations of computation. The historical artifacts include problem texts, diagrams, proofs, tables, formal languages, state models, solver certificates, and machine-checked libraries. The chapters examine how these artifacts make particular relationships available for analysis, what their checks establish, and which assumptions remain outside the check.

The main distinction is between detecting a difference and explaining its origin. A rejected proof step does not by itself refute the theorem. A statistically unusual observation does not identify a defective measurement. A successful calculation does not establish that the calculated model represents the intended problem. We therefore separate the mathematical statement, its representation, the method used to analyze it, and the evidence produced by that method.

Seven parts and 35 chapters are followed by a chronology, a glossary, and an evidence taxonomy. Connections to pattern-oriented software diagnostics provide another way to compare the material. These connections concern analysis operations and artifact relationships. They preserve the differences between a deductive proof, a statistical assessment, an execution trace, and a proposed explanation.

How to read this history

We use diagnostics for investigating a problematic argument, result, representation, or inference. Debugging concerns locating and correcting a defect in a construction, calculation, specification, proof attempt, or implementation. Much of the history predates these terms. The diagnostic interpretation is our present-day reading of documented practices, not the participants' own vocabulary.

In systems theory, observability relates internal states to available outputs under a specified model. Making a proof dependency or an intermediate numerical result inspectable improves the evidence available to an analyst. We call this visibility or inspectability when a mathematical observability property has not been defined.

The boundaries of formal sciences have several classifications. Here we include the mathematical foundations of statistics, computation, control, optimization, and decision theory. Empirical applications explain their use and limitations. Physical instruments, medical diagnosis, and operational software tools belong primarily to the companion histories.

Dates identify texts, announcements, publications, or completed projects. Translations retain their historical setting as well as the translation date; preprints are distinguished from journal publications.

The selected cases represent several traditions without claiming complete coverage or a single line of transmission.

Square brackets identify references. Examples introduced with “consider” or “suppose” are explanatory constructions, not recovered incidents. Connections to our analysis patterns are interpretations or proposed applications. A shared vocabulary does not make an analogy a historical claim.

Neighboring practices and their primary questions

Practice	Primary question	Characteristic evidence
Proof checking	Does the conclusion follow under the specified rules and assumptions?	Derivation, dependencies, proof object
Counterexample search	Is there an admissible instance that violates the claim?	Assignment, structure, trace, boundary case
Model diagnostics	Where does the model fail to account for the relevant evidence?	Residuals, predictive checks, sensitivity
Numerical diagnosis	How do representation and computation affect the result?	Error bounds, conditioning, convergence records
Observability analysis	Can distinct internal states be distinguished from the available outputs?	State model, output map, observation horizon
Debugging	Which change corrects the identified defect?	Reduced case, localized obligation, checked revision

Contents

Seven parts and 35 chapters

PART I

Formal artifacts and the examination of arguments

1. Vocabulary and historical scope

Let us first consider what can go wrong in a formal investigation. We may have an incorrect calculation, an invalid inference, an incomplete case distinction, an inconsistent collection of assumptions, or a model that does not express the intended question. These problems need different artifacts. Repeating an arithmetic operation cannot determine whether the original definition describes the right objects. A proof checker can validate the formal statement supplied to it while the analyst has translated the informal statement incorrectly.

The problem description is therefore part of the evidence. “The proof failed” may mean that the proposition is false, that one proposed derivation is invalid, that the selected automation found no proof, or that a library no longer accepts an old command. “The model failed” may refer to a mathematical inconsistency, poor agreement with data, or an implementation error. Before choosing a method, we need to identify the object and the meaning of failure.

An argument also has several levels of representation. An informal theorem is expressed in ordinary mathematical language. A formal theorem belongs to a particular language with explicit binding, types, and inference rules. A proof script instructs a tool to construct evidence. A proof object records the evidence in a form that a checker can examine. These levels are connected, but they should not be treated as interchangeable. Later proof assistants make some of these connections executable. [1, 2]

We can call the transition between levels a diagnostic boundary. For example, a rational-number model of a computation may satisfy an identity that its floating-point implementation does not preserve. The boundary is the translation from exact arithmetic to finite precision. A failed identity then needs an error analysis, a revised algorithm, or a different specification of acceptable error. The mathematical identity itself may remain valid. [3]

This history follows such boundaries across time. Early texts made algorithms and demonstrations available for repeated inspection. Symbolic languages exposed the form of an inference. Axiomatization made dependencies between assumptions and conclusions more explicit. Computational methods made large searches possible, while proof certificates provided additional means of checking their results.

The resulting development is cumulative and uneven. A modern formalization can still fail because an elementary boundary case was omitted. A long numerical calculation can still require substitution into the original problem. New artifacts add diagnostic possibilities; they do not remove the need to establish what a particular artifact means.

2. Calculation traditions and the preservation of procedures

Written procedures preserve more than answers. They preserve operations that another reader can repeat, compare, and explain. The Nine Chapters on the Mathematical Art presents problems and procedures within a long Chinese mathematical tradition. Liu Hui's commentary, dated to 263 CE, provides explanations that connect procedures with mathematical relationships. The modern translation by Shen, Crossley, and Lun makes both the text and its commentarial setting available for examination. [4]

From a diagnostic viewpoint, a procedure and its explanation answer different questions. Repeating the procedure can identify a transcription or execution error in a particular calculation. Understanding why it works helps determine where the procedure applies and whether a proposed modification preserves its meaning. An answer without either record provides less information when a later calculation differs.

The Indian materials translated by Colebrooke in 1817 include arithmetic and algebraic work attributed to Brahmagupta and Bhāskara. Their problems and rules also show why the history of formal investigation cannot be reduced to one European sequence of symbolic notation. We must retain the distinction between the historical text, the translator's terminology, and our modern algebraic reconstruction. [5]

Al-Khwārizmī's algebraic treatise, composed in the ninth century and translated into English by Rosen in 1831, organized equation solving through verbal rules and geometrical explanations. Modern symbolic notation can make these arguments more familiar, but it can also conceal their original domain restrictions. Historical categories formed around positive quantities should not silently be treated as the unrestricted equations of a later algebra. [6]

Consider a calculation that is checked by substituting its proposed answer into the original problem. This provides a useful separation between producing a candidate and checking it. The check may use a shorter or differently arranged computation. However, if both computations repeat the same mistranscribed input, agreement does not expose that defect. The provenance of the numbers remains relevant even when the arithmetic is reproducible.

A table of worked examples can reveal systematic differences: a repeated correction, an omitted case, or a change in how a quantity is represented. Such patterns are evidence about the calculation process. They are not sufficient by themselves to reconstruct an author's intention. We need the surrounding text and the conventions of the particular tradition.

The preservation of procedures thus creates a practical form of diagnostic memory. A later reader can examine the route to an answer, identify where two versions diverge, and distinguish a changed problem from a changed method. This capability survives the transition from manuscripts to executable notebooks and proof libraries.

3. Geometrical proof and the inspection of dependencies

Euclid's Elements organizes geometrical and arithmetical propositions through definitions, postulates, common notions, and demonstrations. Composed around 300 BCE, it survives through manuscripts and later editions rather than an authorial original. The Bodleian manuscript digitized by the Clay Mathematics Institute was copied in 888 CE. These dates describe different stages of the artifact's history. [7]

For our purposes, the connected arrangement of propositions matters. A reader can ask which earlier result supports a step and what construction makes the required object available. A proof becomes an artifact with dependencies. If an earlier lemma changes, later arguments that use it may need to be reconsidered. We recognize the same general problem in a modern theorem library, although its dependency checks are implemented differently.

A geometrical diagram assists the reader by displaying a configuration. It can also suggest relationships that the assumptions do not guarantee. A point may appear to lie between two others because of its position on the page. A line may look as though it intersects a segment when the construction has not established the intersection. The diagnostic question concerns the inference made from the drawing, not the artistic quality of the drawing.

Hilbert's Foundations of Geometry, published in German in 1899, made five groups of assumptions explicit. Townsend's 1902 translation lists connection, order, parallels, congruence, and continuity, in that order. Hilbert's investigations of consistency and independence let us examine an axiom's relationship to the rest of the system, as well as the steps of a particular proof. [8]

Consider a proposed demonstration that works only when a triangle is acute. If the theorem concerns all triangles, the analysis must account for right and obtuse cases as well. Moving the points in a drawing may reveal the omission. It does not itself supply a proof that the revised argument covers every allowed configuration. Exploration and justification remain separate parts of the work.

Geometrical proof therefore provides an early and continuing example of the distinction between visible arrangement and warranted relationship. In a diagnostic report, we need to say whether we found a misleading representation, a missing premise, or an invalid inference. Each finding leads to a different repair.

4. Algebra analysis and hidden domain assumptions

Algebraic transformations allow a long argument to be compressed into a sequence of expressions. The compression is useful only when the transformations preserve the required relationship. Division assumes that the divisor is nonzero. Taking a square root requires attention to the domain and to the choice of sign. Squaring an equation can introduce candidates that must be checked in the original equation.

Consider the familiar false derivation that begins by setting two quantities equal and later divides by their difference. The written transformation may look ordinary because it has the form of a valid cancellation. The difference is zero under the original assumption. The earliest invalid step can be identified without disputing every later line. This is a small example of localizing a defect by carrying the assumptions through the calculation.

Analysis extended the problem to limits, infinite processes, and functions. Cauchy's Cours d'analyse of 1821 systematically developed notions including limits, continuity, convergence, and infinitesimal quantities. Reading it through a modern annotated translation also shows that historical definitions and arguments should be examined in their own setting. Later terminology does not automatically reproduce every distinction made in an earlier text. [9]

An infinite process introduces a diagnostic boundary that does not occur in the same way in a finite calculation. Properties of every finite stage need not survive a limit. A sequence of continuous functions can have a discontinuous pointwise limit. Additional conditions, such as uniform convergence in the familiar continuity theorem, change what follows. A proof must establish the condition it uses rather than importing it from a picture of several early terms.

As an explanatory example, consider powers of a number in the closed interval from zero to one. For every exponent, the resulting function is continuous. At points strictly below one, the powers approach zero as the exponent grows; at one, they remain one. The pointwise limiting function is discontinuous. The issue lies in exchanging a limiting operation with a property under insufficient assumptions.

These examples show why a mathematical audit needs domains, quantifiers, and modes of convergence. A numerical demonstration of several cases may help locate an error, but it does not restore a missing general condition. A successful repair often produces a revised theorem whose assumptions must be recorded explicitly.

5. Counterexamples and the revision of conjectures

A counterexample is an admissible instance that satisfies the assumptions of a claim and violates its conclusion. Its diagnostic value depends on both conditions. An object outside the stated domain does not refute the theorem. An instance that the calculation has represented incorrectly may reveal a computational defect while leaving the theorem untouched.

Lakatos's Proofs and Refutations examines how proofs, counterexamples, and definitions develop together through a dialogue around the Euler polyhedron formula. The dialogue appeared in four parts in the British Journal for the Philosophy of Science in 1963–1964, before the expanded book of 1976. It is a philosophical reconstruction informed by mathematical history, not a classroom transcript or a literal chronology. [10]

The useful diagnostic distinction is between rejecting a conjecture and locating the assumption that a proof has used without stating. A counterexample can motivate a narrower theorem. It can also show

that a familiar word, such as polyhedron, has been carrying several incompatible meanings. The repair may concern the definition, the statement, or the argument. These are changes to different artifacts.

Consider a conjecture supported by thousands of computed cases. If its first counterexample is extremely large, increasing the search range may eventually locate it. Before that happens, the successful cases establish only that the tested instances passed the implemented check. They do not provide the logical force of a proof over an unbounded domain. The search program and its data representation also need examination.

Reduction can make a counterexample more informative. We can remove irrelevant structure while preserving the violated property, or compare a failing instance with a nearby passing one. The resulting smaller artifact may expose an omitted boundary condition. Minimality, however, always concerns the allowed reduction operations. A case minimal under deletion need not be smallest under every possible representation.

Modern property testing and failure reduction make these activities executable. QuickCheck, described by Claessen and Hughes in 2000, generates tests from properties; Zeller and Hildebrandt's 2002 delta debugging work systematically reduces failure-inducing input. Their software context differs from mathematical refutation, but the shared analysis operation is clear: preserve a discriminating failure while removing unnecessary detail. [11, 12]

We should also preserve unsuccessful conjectures. They record which generalizations failed and why. A collection containing only repaired statements loses part of the evidence needed to understand the boundaries of the resulting theory.

PART II

Logical foundations and limits of decision

6. Symbolic logic and explicit inference

Logical analysis long predates nineteenth-century symbolic languages. Aristotle's Prior Analytics examined forms of syllogistic inference in the fourth century BCE. Its distinction between the premises supplied and what follows from their arrangement remains relevant to diagnosis. A conclusion can be true even when a proposed inference is invalid; a valid inference can begin from false premises. Checking the form and checking the premises are different investigations. [13]

Symbolic logic makes aspects of an argument available for manipulation independently of its particular subject matter. Boole's An Investigation of the Laws of Thought, published in 1854, developed an algebraic treatment of logical relationships. Its notation and interpretation belong to the work itself; they should not be identified without qualification with every later presentation of Boolean algebra. [14]

Frege's Begriffsschrift of 1879 introduced a formal language designed to represent inferential relationships with a precision unavailable in ordinary mathematical prose. Quantification and the structure of nested assertions became explicit parts of the representation. This provided another way to examine what an argument says before deciding whether its steps are valid. [15]

The diagnostic significance of quantification can be seen without elaborate notation. "For every input there is a suitable output" allows the output to depend on the input. "There is one output suitable for every input" makes a different claim. A proof that changes the order of these commitments can appear persuasive in prose while proving the wrong statement.

Negation introduces related problems. The failure to establish a proposition is not a proof of its negation. In an automated search, the absence of a proof after a fixed interval is an observation about that search. It may reflect a poor strategy, a missing lemma, insufficient resources, or the nature of the logic. The artifact needs a result status that preserves these alternatives.

Formalization can also reveal an ambiguity that was harmless in one context and consequential in another. A variable name may be reused with a different scope. An existential witness may depend on assumptions that are later discharged. A definition may use a term before its meaning has been fixed. These are structural defects in the representation of the argument.

The benefit of symbolic language is therefore not that symbols remove interpretation. They make selected interpretive commitments explicit and subject to rules. The translation from an informal problem to that language remains a separate task. A formally valid derivation of an incorrectly translated statement can still fail the original purpose of the investigation.

7. Axioms paradoxes and the repair of foundations

An axiom system identifies the assumptions from which a body of results is developed. It allows us to ask whether a proposed object exists under those assumptions, whether one assumption follows from others, and whether the system permits a contradiction. The diagnostic object is now the collection of rules and commitments, not only an individual calculation.

Russell's The Principles of Mathematics, published in 1903, discusses the contradiction associated with considering classes that are not members of themselves. In modern set notation, the unrestricted construction can be displayed as follows. [16]

$$R = \{x \mid x \notin x\}$$

If the proposed collection is a set and membership is interpreted in the unrestricted way, asking whether it belongs to itself produces contradictory conditions. The problem does not show that every collection is contradictory. It exposes the incompatibility of particular formation assumptions with unrestricted self-application.

Zermelo's 1908 axiomatization offered a different framework for set formation. In particular, forming a subset by a property is tied to an already given set rather than granting a set for every unrestricted property. This is one historical response to foundational problems; later set theories developed further principles and different formulations. [17]

A restriction that blocks one contradiction must itself be examined. What constructions remain available? Which earlier proofs need reformulation? Does the restriction address the actual source of inconsistency, or merely prevent one way of writing it? These are questions about the consequences of a repair. Removing the visibly failing expression is not a sufficient account of the resulting theory.

The same general issue appears in formal specifications. Suppose one requirement permits a resource to have exactly one owner while another requires two distinct exclusive owners at the same time. No implementation can satisfy both requirements under their ordinary formalization. Debugging the implementation alone cannot resolve the conflict. The assumptions must be revised or their intended meanings clarified.

Foundational diagnostics also requires restraint. A contradiction discovered in one formal system does not discredit every use of mathematics. Conversely, the absence of a known contradiction is not a general proof of consistency. We need to state the system, the relevant assumptions, and the kind of consistency evidence being offered.

8. Proof theory and the structure of a derivation

A proof can be studied as a mathematical object. Its inference steps have a structure, and transformations of that structure can be investigated. Gentzen's 1935 work on logical deduction

introduced natural deduction and sequent calculi, giving systematic forms to reasoning with assumptions and conclusions. [18]

This changes what can be inspected. Instead of reading only a succession of asserted propositions, we can examine where an assumption enters, how it is used, and where it is discharged. A proof tree makes some dependency relationships explicit. A sequent records the assumptions available for a conclusion. A local error can then be described in terms of the permitted inference rule and its premises.

Cut elimination, in the appropriate calculi, concerns the removal of an intermediate inference mechanism while preserving derivability. Its diagnostic interest is that a derivation can sometimes be reorganized into a form that exposes its dependence on the formulas at issue. This does not mean that every transformed proof is shorter or easier for a human reader. Normalization can increase the size of the explicit artifact.

Consider an argument that uses a lemma proved under a temporary assumption. If that assumption is omitted when the lemma is later applied, the final conclusion may look supported while one dependency has been lost. A proof representation that records contexts allows this mismatch to be checked mechanically. The error is in the movement of evidence between contexts.

Proof theory also distinguishes different logical systems. Classical and intuitionistic derivations do not permit exactly the same principles. A translation between them must be specified. An argument acceptable in one framework may require a different conclusion or additional construction in another. A diagnostic report should name the framework instead of describing all rejected steps as universally invalid.

For the later history of proof assistants, this explicit treatment of derivations provides both opportunity and cost. A tool can check many small steps reliably, but the analyst must supply definitions and evidence in a compatible form. The resulting proof object is an artifact of a particular logic and implementation. Its acceptance is meaningful because those commitments are sufficiently explicit to be examined.

9. Incompleteness computability and limits of diagnosis

Hilbert’s 1900 address placed foundational problems alongside questions from geometry, number theory, and analysis. His second problem concerned the compatibility of the arithmetical axioms. The English publication of the address appeared in 1902. The distinction between the occasion and the printed translation is useful when following later responses. [19]

Gödel’s 1931 work established limits on sufficiently strong, effectively axiomatized formal systems under specified consistency assumptions. Incompleteness concerns what such systems can prove; the second incompleteness theorem further restricts internal proofs of consistency under the relevant conditions. These results should not be converted into a general statement that every mathematical question is undecidable or that formal checking has no value. [20]

Church's 1936 paper and Turing's work of 1936–1937 gave precise forms to limitations of effective decision procedures. Turing's machine model made the operations of a computation explicit. The resulting undecidability concerns uniform algorithms over a specified class of problems. It does not prohibit proofs about particular programs or decision procedures for restricted classes. [21, 22]

Several diagnostic statuses must therefore remain separate. A statement may have a proof, have a counterexample, be independent of a stated axiom system, or simply remain unresolved in the current investigation. A tool may return a timeout, exhaust its search bound, or fail because of an implementation problem. The last group of outcomes does not establish a mathematical impossibility result.

Consider an analyzer that examines arbitrary programs and promises always to decide whether they terminate. The general promise conflicts with the relevant undecidability result. Restricting the input language or accepting an "unknown" result changes the promise. Such a method can still be useful, provided its scope and result statuses are preserved.

These distinctions also protect comparisons across disciplines. Missing measurements, unidentifiable parameters, computational expense, and logical undecidability can all prevent an answer, but they arise from different structures. Adding a sensor may resolve the first problem. A stronger computer may address some resource limitations. Neither action is a general remedy for an impossibility theorem.

10. Complexity certificates and bounded investigation

Decidability does not establish practical feasibility. A decision procedure may terminate on every input while requiring resources that grow too rapidly for the instances of interest. Complexity theory studies such resource requirements under specified computational models. It gives another level at which an unsuccessful investigation can be explained.

Cook's 1971 paper on the complexity of theorem-proving procedures established a central completeness result connecting propositional satisfiability with nondeterministic polynomial-time computation. For diagnostic purposes, the distinction between finding a solution and checking a proposed solution is especially useful. A satisfying assignment can be checked against a finite Boolean formula even when locating it was difficult. [23]

The negative answer requires separate consideration. A solver that finds no satisfying assignment before it is stopped has not established unsatisfiability. A complete search or an appropriate refutation certificate can support that conclusion. The logical force lies in the checked evidence, not in the amount of time spent or the size of the search log.

Bounds make many otherwise difficult investigations manageable. We can restrict the number of objects in a relational model, the depth of a trace, or the magnitude of an integer. A counterexample inside the bound can refute an unrestricted universal claim if the encoding faithfully represents it. The absence of a counterexample inside the bound usually establishes only a bounded result.

Alloy's relational modeling approach, described by Jackson in 2002, made this distinction central to a practical analysis method. Small instances can expose surprising specification defects. Their value depends on knowing the scope that was searched and the translation used to represent the model. A successful bounded check should not be reported as an unrestricted theorem without an additional completeness argument. [24]

A useful result therefore includes the input, scope, assumptions, resource outcome, and any checkable witness. These details allow another analyst to reproduce the search or understand why a different search produced a different result. Complexity does not merely limit automation; it also explains the importance of certificates, decomposition, and carefully chosen abstractions.

PART III

Probability statistical diagnosis and numerical evidence

11. Probability and the formalization of uncertain inference

Probability theory provides a mathematical language for uncertainty, but a probability model is still an assumption about the relevant possibilities. We need to specify what can occur, which events are being considered, and how probabilities are assigned. A correct calculation within the model does not establish that these assignments describe the intended situation.

Bayes's essay, communicated by Richard Price and published in 1763, addressed an inverse probability problem. Its historical formulation should be distinguished from the many later methods described as Bayesian. The general relation now called Bayes's rule connects a prior probability with a likelihood and a posterior probability. The conditioning information belongs to the statement of the calculation. [25]

$$P(H \mid E) = \frac{P(E \mid H)\, P(H)}{P(E)}$$

Here H denotes a hypothesis and E denotes evidence within a specified probability model. The denominator must be positive. For continuous quantities, the corresponding calculation requires appropriate probability densities or conditional distributions; point probabilities cannot simply be substituted without considering the model.

Kolmogorov's 1933 axiomatization formulated probability in a measure-theoretic setting. This gave probability a mathematical foundation that could accommodate finite and infinite spaces under explicit rules. The framework establishes relationships among probabilities; interpreting a particular model as frequencies, beliefs, or a representation of a physical process requires further commitments. [26]

Fisher's 1922 paper on the mathematical foundations of theoretical statistics investigated estimation through concepts including consistency, efficiency, and sufficiency. These concern the information and behavior of statistical procedures under models. They provide criteria for comparing methods that go beyond whether one fitted value looks reasonable. [27]

Consider a diagnostic test for a rare condition. Even a test with good sensitivity and specificity can produce a substantial proportion of false positives among its positive results when the condition is sufficiently rare. The probability of a positive result given the condition and the probability of the condition given a positive result are different quantities. Reversing them changes the question.

Formal probability thus supplies both an inference method and checks on the interpretation of its output. A diagnostic analysis must preserve the population, sampling mechanism, conditioning information, and model assumptions. Without these, a numerically precise probability can communicate less reliable information than a clearly stated unresolved alternative.

12. Residuals influence and the diagnosis of fitted models

Fitting a statistical model produces estimates. It also produces artifacts that can be examined for discrepancies between the fitted relationships and the observations. Residuals are one such artifact. Their magnitude, arrangement, and relationship to explanatory variables can show where the fitted model requires further investigation.

A residual is not automatically a measurement error. It can contain random variation allowed by the model, an omitted relationship, a recording error, or the effect of an unsuitable functional form. Interpreting it requires the original observations and the model that produced it. A residual plot without that context can encourage an attractive but unsupported explanation.

Cook's 1977 paper introduced a measure of the influence of an observation in linear regression through its effect on the fitted result. Influence differs from having a large residual. An observation in an unusual position in the predictor space can materially affect a fitted relationship even when its residual is not the largest. The distinction changes which artifacts need inspection. [28]

Suppose a straight line is fitted to data generated by a curved relationship. A residual plot can show systematic curvature even when the fitted line explains a large part of the overall variation. Removing the most visibly unusual observation may improve a summary statistic while leaving the structural mismatch. The repair belongs to the model, unless there is separate evidence that the observation itself is defective.

Box's 1976 discussion of science and statistics emphasized the relationship between model construction and criticism. In this setting, a model is a means of investigation whose adequacy depends on the purpose and the discrepancies that matter. Diagnosing one limitation does not imply that every use of the model becomes invalid. [29]

We can compare several views of the same fitted model: residuals against fitted values, residuals against time, sensitivity to influential observations, and predictions in relevant regions. Each view can reveal a different relationship. Agreement across several plots is useful, although it does not prove that all unexamined relationships are satisfactory.

The diagnostic result should therefore describe a discrepancy and its implications before prescribing deletion or refitting. A large residual is an observation. "This point is wrong" is a hypothesis about its provenance. Keeping those statements separate preserves evidence that might otherwise be discarded during an automatic cleaning step.

13. Hypothesis tests multiplicity and error control

Statistical testing formalizes a decision or evidential question relative to a probability model. It requires a test statistic, a reference distribution, and a rule for interpreting the result. The numerical output has no single meaning independent of those choices.

Neyman and Pearson's 1933 work considered the construction of efficient tests by examining error probabilities and alternatives. The probability of rejecting a true null hypothesis and the probability of failing to reject a specified false null hypothesis are properties of a procedure under repeated sampling assumptions. They are not automatically posterior probabilities that the hypotheses are true or false. [30]

A p-value concerns results at least as incompatible with the null model, according to the chosen statistic, as the result observed. It does not give the probability that the null hypothesis is true. A small value can motivate investigation, but the practical meaning also depends on effect size, study design, and whether the model and analysis choices were appropriate.

Multiplicity changes the diagnostic setting. If many independent true null hypotheses are each tested at the same nominal level, the probability of at least one rejection grows with the number of tests. Searching across outcomes, transformations, subgroups, or models can create related selection effects even when the final report contains only one result.

Benjamini and Hochberg's 1995 procedure addressed false discovery rate control. Its original guarantees depend on assumptions about the tests, including an independence setting; later work developed other conditions and variants. False discovery rate is the expected proportion of false rejections among rejections, with the proportion taken as zero when there are none. It differs from the probability of making any false rejection. [31]

For diagnosis, we need the analysis history as well as the final statistic. Was the test selected before inspecting the outcome? How many comparisons were considered? Did a failed attempt lead to a change in the hypothesis or merely a different presentation of the same data? The sequence of decisions affects the interpretation of the reported evidence.

The connection to trace analysis is methodological. A final result is one event in a longer sequence of data preparation, model choice, testing, and revision. Preserving that sequence can expose why two apparently identical analyses disagree. It also prevents a selected success from being presented as though it were the only attempted investigation.

14. Bayesian model checking and computational calibration

Bayesian inference produces a posterior distribution conditional on a model, prior assumptions, and observations. This does not remove the need to examine the model. A posterior can be calculated accurately for a model that describes the data poorly. Conversely, a useful model can be implemented by an unreliable computational method.

Gelman, Meng, and Stern's 1996 work on posterior predictive assessment compared features of observed data with corresponding features of data generated through the fitted model. The chosen discrepancy determines which aspects of fit are examined. A check of the mean can miss a problem in the tails; a check of marginal distributions can miss a dependence structure. [32]

Computational diagnosis asks another question. Markov chain Monte Carlo generates dependent draws intended to explore a target distribution. A chain can remain in one region, mix slowly, or fail to reveal a separated mode. A visually smooth trace is not sufficient evidence that the relevant distribution has been explored.

Vehtari and colleagues' 2021 treatment of rank normalization, folding, and localization improved familiar convergence diagnostics and effective sample size assessments. These are tools for finding problems in finite simulation output. Passing them does not constitute a mathematical proof that every feature of an unknown target distribution has been estimated accurately. [33]

Simulation-based calibration, described by Talts and colleagues in a 2018 preprint revised in 2020, uses repeated simulated parameter and data draws to examine an inference procedure. The expected rank behavior under the generative setup provides a check on the relationship between simulation and posterior computation. The procedure can reveal implementation or inference problems without requiring a known closed-form posterior for every test case. [34]

The distinction between these checks is consequential. Posterior predictive checking examines aspects of model fit. Chain diagnostics examine the behavior of a sampling computation. Simulation-based calibration examines calibration across a specified simulation experiment. A successful result in one category should not be substituted for a missing check in another.

There is also a possible common defect. If a simulator and an inference implementation share the same erroneous convention, their agreement can conceal a mismatch with the intended model. Independent formulations, simple special cases, and explicit definitions remain useful. The mathematical model, its implementations, and the observed application need separate lines of evidence.

15. Numerical error conditioning and validated computation

Numerical analysis investigates the relationship between exact mathematical problems and their finite computations. The distinction is essential when a result looks plausible but changes substantially with precision, algorithm, or input perturbation. We need to determine whether the problem is sensitive, the algorithm is unstable, or the implementation is defective.

Wilkinson's Rounding Errors in Algebraic Processes, published in 1963, developed systematic analysis of finite arithmetic. Backward error analysis asks whether the computed result solves a nearby problem exactly. Forward error concerns its difference from the desired exact answer. These questions connect the observed output to both the algorithm and the sensitivity of the original problem. [35]

Conditioning belongs to the problem and its chosen input-output representation. Stability belongs to the numerical method. A stable method can produce a large forward error on an ill-conditioned problem because small data perturbations already have a large effect. Changing the algorithm cannot remove sensitivity intrinsic to the information supplied. [3]

Goldberg's 1991 account of floating-point arithmetic explains why finite representations require rules for rounding and exceptional values. Arithmetic operations that are algebraically interchangeable over real numbers can give different finite results. Cancellation can remove leading significant digits; evaluation order can change accumulated rounding. Such behavior needs interpretation against the actual arithmetic model. [36]

Consider a subtraction of two close approximations. Each approximation may have a small relative error, while their difference has a large relative error. Printing more digits of that difference does not recover the information already lost. A reformulated expression, additional precision, or a justified error bound may be needed.

Interval analysis provides another kind of artifact: an enclosure intended to contain the exact result under stated input and rounding assumptions. Moore, Kearfott, and Cloud's 2009 treatment develops methods for computing and using such enclosures. Repeated use of a dependent quantity can widen an interval substantially; a wide enclosure is not itself evidence that the underlying quantity varies over the entire enclosed range. [37]

For a numerical investigation, useful supporting information includes the algorithm, precision, stopping criterion, conditioning assessment, and independent checks. Agreement between two implementations is stronger when their likely failure mechanisms differ. A common library, common formula, or common data conversion can preserve the same defect in both results.

PART IV

Information internal state and constrained solutions

16. Information theory redundancy and error syndromes

Information theory made it possible to analyze communication in terms of uncertainty, coding, and channel behavior. Shannon's 1948 paper separated the mathematical problem of communication from the semantic interpretation of a message. That separation is important when we later use information measures in diagnostics: a measure of symbol uncertainty does not by itself measure the usefulness or truth of the message. [38]

Redundancy can support error detection because valid encoded messages occupy only part of the space of possible received messages. A received word outside that set indicates a discrepancy relative to the code. Error correction needs a further property: the encoding and error assumptions must allow the receiver to identify an intended valid word.

Hamming's 1950 paper developed error-detecting and error-correcting codes using systematic relationships among bits. A syndrome records which parity constraints are violated. Under suitable assumptions, such as a specified single-bit error model, the pattern can identify a correction. If the actual corruption exceeds the code's assumptions, the same mechanism can become ambiguous or lead to an incorrect correction. [39]

Consider a single parity bit added to a message. It detects any odd number of bit flips in the protected word, but it does not detect every even number. A passing parity check therefore has a precise scope. It means that the checked parity relationship holds, not that the entire transmission is known to be unchanged.

The diagnostic analogy concerns designed constraints. A checksum, invariant, or consistency relation can make selected corruptions detectable. Its blind spots depend on the relationship it preserves and the error model it was designed to discriminate. We should not infer arbitrary defect mechanisms from one failed check.

Information measures also require a reference distribution and a sampling unit. An increase in trace-message entropy may reflect varied activity, changed formatting, noise, or a different mixture of workloads. The number alone does not select among those explanations. A diagnostic interpretation combines the statistic with message meaning and the process that produced the trace.

Coding theory thus supplies a particularly clear example of evidence designed in advance. The system includes relationships that later observations can check. Its history helps explain why diagnostic capability depends on what distinctions the representation preserves before a failure occurs.

17. Mathematical observability and state reconstruction

Observability in systems theory asks whether available outputs distinguish internal states under the specified dynamics and inputs. The question concerns a relationship within a model. It does not reduce to the number of recorded variables or the volume of telemetry.

Kalman introduced observability and controllability, together with their duality, in his contribution to the first IFAC Congress in Moscow in 1960; the proceedings appeared in 1961. His 1963 paper developed these concepts within a systematic treatment of state-space realizations. The introduction and the later treatment are distinct historical steps. [40, 41]

For a finite-dimensional, discrete-time, linear time-invariant system, one familiar form is the following. [41]

$$x_{k+1} = Ax_k + Bu_k; \quad y_k = Cx_k$$

Here x_k is the state, u_k the known input, and y_k the measured output at step k. The matrices A, B, and C describe state evolution, input action, and output measurement. State reconstruction concerns whether different initial states can produce the same output sequence. For a state dimension n, the standard observability matrix stacks the output matrix and its successive products with the state matrix.

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}; \quad \operatorname{rank}(\mathcal{O}) = n$$

Full column rank gives observability in this setting. The assumptions matter: the result stated here is not automatically the criterion for an arbitrary nonlinear, time-varying, noisy, or infinite-dimensional system. Numerical conditioning of reconstruction can also be poor even when the exact rank criterion is satisfied.

Consider a two-component state whose components remain constant and whose only measured output is their sum. Different pairs with the same sum produce the same observations. Repeating that measurement does not reveal how the sum is divided. An additional independent output can distinguish the components; a longer record of the same unchanged sum cannot.

Kalman's 1960 filtering paper concerns estimation in a probabilistic dynamical setting. Estimation and observability are related, but they answer different questions. An estimator uses model and uncertainty assumptions to compute a state estimate. Those assumptions do not turn indistinguishable states into uniquely observed facts. [42]

Willsky's 1976 survey examined failure detection in dynamic systems, including approaches using analytical relationships and filtering. Here a discrepancy between predicted and measured behavior becomes a diagnostic signal. Detection and isolation still differ: several faults may affect the same residual. [43]

This gives a precise connection to practical instrumentation. Before collecting more data, we can ask which candidate states or faults the proposed outputs could distinguish. The answer can justify an additional observation channel or show that the current diagnostic question exceeds what the model and measurements determine.

18. Identifiability inverse problems and regularization

Observability commonly concerns states under a specified model. Identifiability concerns whether model parameters can be distinguished from the permitted observations or experiments. The distinction prevents a fitted parameter value from being mistaken for a uniquely established mechanism.

Bellman and Åström's 1970 paper introduced structural identifiability as a central question in parameter identification. Structural analysis concerns distinguishability under idealized information and the chosen model. Practical identifiability additionally concerns finite, noisy, or poorly informative data. A parameter may be structurally identifiable while remaining difficult to estimate in a particular experiment. [44]

Suppose the output depends only on the product of two unknown positive parameters. Many pairs share the same product. A numerical optimizer may return one pair and report a small residual, but the output alone does not distinguish that pair from the alternatives. The flat direction belongs to the model-data relationship. It is not necessarily a defect in the optimizer.

Inverse problems ask for causes, parameters, or structures from their consequences. Existence, uniqueness, and stability are separate questions. A solution may exist without being unique. A unique solution may depend so sensitively on the data that small perturbations produce large changes. These differences determine what sort of diagnosis is possible.

Tikhonov's 1963 work on regularization developed methods for treating incorrectly formulated, or ill-posed, problems by incorporating additional structure. In a familiar modern form, an objective balances agreement with the observations against a penalty or constraint on the solution. The selected result then depends on the regularization choice as well as the observations. [45]

Regularization should therefore be reported as part of the inference. A smooth reconstruction is not evidence that the unobserved object was smooth unless that assumption is independently supported. A sparse estimate may be useful for a purpose while other nonsparse explanations remain compatible with the data.

Diagnostic progress can come from changing the experiment rather than adding computation. A different input, observation time, or output variable can distinguish parameters that the original setup leaves inseparable. Sensitivity analysis helps identify which changes are likely to matter, although its local conclusions must not be confused with a global uniqueness result.

The general principle is simple: preserve the alternatives that the evidence does not distinguish. A single plotted reconstruction is only one artifact. The family of compatible reconstructions, the uncertainty, and the assumptions used to select among them may contain the more useful diagnostic information.

19. Optimization feasibility and certificates

Optimization asks for a best admissible solution according to a specified objective. The word admissible already contains a diagnostic question: are the constraints compatible? If they are not, the search for a better objective value cannot succeed until the formulation changes.

Kuhn and Tucker's contribution to nonlinear programming appeared in the 1951 proceedings of the second Berkeley symposium, held in 1950. The optimality conditions associated with this line of work relate derivatives, constraints, and multipliers. Their use requires appropriate hypotheses. Stationarity alone does not establish a global optimum for an arbitrary nonconvex problem. [46]

Duality provides ways of comparing a candidate objective value with a bound. Under the relevant feasibility and duality conditions, primal and dual evidence can certify optimality or constrain the remaining gap. Boyd and Vandenberghe's Convex Optimization develops these relationships in a setting where the assumptions and resulting guarantees can be stated precisely. [47]

The infeasible case produces another useful artifact. Chinneck and Dravnieks's 1991 work examined minimal infeasible constraint sets in linear programs. A small incompatible subset can make a formulation error easier to understand. An irreducible set is one in which removing any member restores feasibility for that subset; it need not have the smallest possible cardinality among all incompatible subsets. [48]

Consider a planning model that requires a task to consume at least five units of a resource while allowing at most three units in total. The contradiction is immediate once the relevant constraints are isolated. In a larger model, these limits may be distributed across calendars, unit conversions, and policy rules. The solver's infeasibility result becomes useful when it can be connected back to those meanings.

A numerical certificate also needs appropriate interpretation. Floating-point tolerances can permit small constraint violations or conceal near-degeneracy. Scaling can affect a solver's behavior. The mathematical optimization problem and the numerical termination criterion are separate records, even when a tool displays one concise status message.

An "optimal" solution can also fail its intended purpose if the objective is wrong. Minimizing an average can conceal unacceptable extremes; minimizing cost can omit a required constraint. Formal optimality is conditional on the formulated problem. Diagnosis must examine the formulation as well as the solver and its output.

20. Decision theory and the interpretation of solutions

Formal decision problems specify alternatives, outcomes, and a criterion for comparison. Game theory adds interactions among decision makers whose outcomes depend on one another's choices. These representations can expose incompatible expectations, but they do not remove the need to explain what a mathematical solution means in its application.

Nash's 1950 note established the existence of an equilibrium for finite games when mixed strategies are allowed. At such an equilibrium, no player can improve that player's expected payoff by a unilateral deviation while the other strategies remain fixed. Existence is a statement about the specified game. It does not guarantee a unique equilibrium, a particular learning process, or the best collective outcome. [49]

This separation is useful in diagnostics. A computed strategy profile can satisfy an equilibrium condition while failing the designer's goal. The computation may be correct and the modeled incentives may explain the unwanted behavior. Changing the numerical solver would then leave the underlying relationship unchanged.

Graph and constraint representations offer another view of decision problems. A feasible assignment requires compatible relationships among tasks, resources, and permissions. A graph can make a forbidden cycle or an unavailable assignment visible. However, the graph must represent the intended relationship. A reachability edge, a causal edge, and a resource-conflict edge are not interchangeable merely because they can all be drawn as arrows.

Consider two independently optimized components that share a capacity limit. Each local solution can be feasible under its local assumptions, while their composition exceeds the common capacity. The defect lies in the interface between models. Adding the shared constraint changes both the feasible set and the meaning of the local optima.

Causal diagrams provide a related but distinct mathematical structure for expressing assumptions about interventions and dependencies. Pearl's 1995 paper developed rules for identifying causal effects from such structures. A causal conclusion depends on the graph and associated assumptions; an observed correlation alone does not supply them. [50]

These examples show why the formal sciences belong in a history of diagnostics beyond numerical calculation. They allow us to inspect the architecture of a decision: what the participants know, what they can change, which constraints they share, and what the selected outcome certifies. The remaining judgment concerns whether that architecture represents the problem we intended to solve.

PART V

Programs specifications and automated reasoning

21. Semantics types and the meaning of a program

A program is an executable artifact and a mathematical object under a specified semantics. To reason about its behavior, we need to state what its expressions, commands, and interactions mean. Without that connection, a proof can concern a formal object that differs from the program actually executed.

Scott and Strachey's 1971 report Toward a Mathematical Semantics for Computer Languages developed a mathematical treatment of program meaning. Compositional descriptions connect the meaning of a compound construction with the meanings of its parts. This provides a basis for asking whether a transformation preserves the relevant behavior. [51]

Different semantic descriptions expose different evidence. An operational account can display transitions between states. A denotational account associates a construction with a mathematical meaning. An axiomatic account states relationships that hold before and after execution. These approaches can support one another, but their correspondence must be established rather than assumed from shared notation.

Types make another set of distinctions explicit. Milner's 1978 paper on type polymorphism developed a type discipline and inference method for a language setting with reusable polymorphic functions. The guarantees belong to the defined language and type system. A typing result does not establish every desired program property, such as termination, absence of resource exhaustion, or conformity with an external requirement. [52]

Consider a function that accepts a temperature and produces a distance because both quantities are represented by floating-point numbers. A basic numerical type may not distinguish the mismatch. A richer representation can preserve the units or roles and reject the operation. The diagnostic capability depends on which distinctions the type system expresses.

A type error can also be reported far from its origin. An earlier inference may impose a constraint that becomes inconsistent only when another expression is encountered. The displayed location identifies a point where the accumulated constraints fail; it need not identify the intended correction. Explanations that expose the constraint path can improve the usefulness of the error message.

Semantics and types therefore extend diagnostics before execution. They make selected classes of incompatibility visible in a representation. Their limitations are equally informative. A property omitted from the formal language remains outside the guarantee, even when all represented obligations have been checked.

22. Program logic invariants and abstract interpretation

Reasoning about programs requires relationships that survive execution steps. Floyd's 1967 work on assigning meanings to programs associated assertions with program structure. Hoare's 1969 axiomatic basis for programming expressed how a command relates a precondition to a postcondition. These developments made program correctness obligations explicit mathematical objects. [53, 54]

$$\{P\}\ C\ \{Q\}$$

Here P is the precondition, C the command, and Q the postcondition. In its partial-correctness reading, the assertion says that if the precondition holds and the command terminates, the postcondition holds. Termination needs a separate argument for total correctness. Omitting this distinction can make a proof of a nonterminating program look like a demonstration that the program will produce a result.

A loop invariant is a property preserved by each iteration when its proof obligations are satisfied. It must hold initially and be strong enough, together with loop exit, to support the required conclusion. A plausible formula that happens to hold in several observed iterations has not thereby been established as an invariant.

Consider a loop that sums a list. An invariant may relate the accumulated sum to the portion already processed. A boundary error can break that relationship in the initialization, the update, or the exit condition. Examining these obligations separately localizes the problem more effectively than treating the loop as one opaque object.

Cousot and Cousot's 1977 abstract interpretation framework organized sound approximation of program behavior through relationships between concrete and abstract domains. An analyzer can reason about intervals, signs, or other properties while representing many concrete states together. The gain in tractability is accompanied by a possible loss of precision. [55]

A sound overapproximation can include behaviors that no concrete execution exhibits. A warning may therefore identify a real problem or a limitation of the abstraction. Dismissing every warning as noise would be unsound practice, but treating every warning as a demonstrated execution failure would also misstate the result.

The diagnostic task is to connect the abstract result back to the concrete semantics and the intended property. Refining the domain or adding a relevant invariant can distinguish alternatives that the earlier approximation merged. A more precise analysis is useful when it resolves a concrete ambiguity, not merely because its internal representation is larger.

23. Temporal logic model checking and counterexample traces

Many properties concern sequences rather than isolated states. A request should eventually receive a response. Two activities should never occupy a critical section together. A resource should become available again after use. These statements involve different relationships over time.

Pnueli's 1977 paper applied temporal logic to programs. The development of model checking by Clarke and Emerson and by Queille and Sifakis in the early 1980s made systematic examination of finite transition systems a practical form of formal analysis. The Clarke–Emerson workshop contribution dates to 1981, with proceedings published in 1982; the CESAR paper appeared in 1982. [56, 57, 58]

For a safety property, a counterexample often takes the form of a finite path to a forbidden state. Liveness failures may require an infinite behavior represented by a finite prefix and a repeating cycle. Some branching-time properties require richer counterexample structures. We should not assume that every formal failure can be explained by one ordinary linear log.

A model-checker trace is generated from the model. It is not automatically a recording of a physical or software execution. The correspondence between modeled transitions and actual operations must be examined before the trace is used as an incident explanation. An environmental behavior permitted by the model may be impossible in the deployed system, or an omitted behavior may be possible there.

Counterexample-guided abstraction refinement, described by Clarke and colleagues in 2000, makes this distinction part of an iterative process. A reported abstract counterexample is analyzed to determine whether it represents concrete behavior. A spurious counterexample motivates refinement of the abstraction; a concrete one supports investigation of the property or system. [59]

Fairness assumptions are another diagnostic dependency. A liveness argument may assume that an enabled activity will eventually execute. Removing or changing that assumption can change the result without changing the transition rules. The assumptions therefore belong with the trace and the property, rather than remaining hidden in a tool configuration.

Model checking extends the available evidence from selected test executions to systematic exploration under a finite or otherwise manageable representation. Its value depends on preserving the state space, property, abstraction, and assumptions that gave the result its meaning.

24. Automated deduction SAT and satisfiability modulo theories

Automated reasoning turns logical structure into a search problem. Davis and Putnam's 1960 procedure for quantification theory and Robinson's 1965 resolution principle supplied influential foundations for this development. They show how logical consequences can be generated through explicitly defined operations on formulas. [60, 61]

Propositional satisfiability asks whether truth values can be assigned so that a Boolean formula holds. Satisfiability modulo theories extends the setting with interpreted structures such as arithmetic, arrays,

or equality. Nieuwenhuis, Oliveras, and Tinelli's 2006 account of SAT and SAT modulo theories developed a framework connecting Boolean search with theory reasoning. [62]

The diagnostic artifacts differ by outcome. A satisfying assignment provides a candidate model that can be checked against the encoded constraints. An unsatisfiability proof can support the claim that no such assignment exists. An unsatisfiable core identifies a conflicting subset, but a core need not be minimal unless that property has also been established.

An encoding error can invalidate an application even when the solver answers the encoded question correctly. A missing constraint can admit a spurious solution. An extra constraint can make a feasible intended problem appear impossible. The translation from the original problem is therefore an independent boundary for investigation.

Proof logging helps separate search from checking. Wetzler, Heule, and Hunt's 2014 DRAT-trim work provided checking and trimming for an expressive format of clausal proofs. A solver can use sophisticated search methods while emitting evidence for a separate checker. The checker still relies on a correct understanding and implementation of the proof format. [63]

The 2016 solution of the Boolean Pythagorean triples problem by Heule, Kullmann, and Marek used SAT solving and refutation evidence separately checked by the DRAT-trim proof checker. The mathematical question concerned a two-coloring of positive integers avoiding monochromatic Pythagorean triples. The work illustrates how a finite encoded obstruction can settle an unbounded statement when the connection to the original problem is justified. [64]

Large machine-generated evidence also creates a practical diagnostic problem. Reproducing the entire search may be expensive, while identifying which part of a proof explains the obstruction may be difficult. A certificate can establish a result without providing a concise human explanation. Both artifacts have value, but they answer different questions.

25. Model based diagnosis and competing explanations

Model-based diagnosis starts with a description of expected behavior and observations that may conflict with it. Instead of associating a symptom with one remembered cause, it asks which assumptions about components or mechanisms must be reconsidered to restore consistency.

Reiter's 1987 theory of diagnosis from first principles formalized this approach. A system description, observations, and assumptions about normal component behavior determine conflicts and candidate diagnoses. Minimal diagnoses are related to hitting sets of conflicts. This is a precise relationship within the formal theory, not a general guarantee that the smallest diagnosis names the actual physical cause. [65]

Suppose a small circuit model predicts that an output must be high while a reliable observation says it is low. Several explanations may be possible: one component behaves abnormally, an input assumption is

wrong, or the observation is incorrect. If the formal diagnosis framework permits only component abnormalities, it will search only that space of explanations.

This limitation matters because the candidate vocabulary shapes the result. A real failure omitted from the model cannot be selected as such. It may be represented indirectly as an abnormality elsewhere or leave the observations unexplained. Extending the model changes the diagnostic question and may change the set of minimal candidates.

Additional observations are useful when they discriminate among remaining diagnoses. Repeating a measurement predicted identically by all candidates may add confidence in the observation without helping isolation. A different test can split the candidate set. Selecting it requires information about predicted outcomes and, in practical settings, the cost or disturbance of the test.

Minimality also needs interpretation. A subset-minimal diagnosis has no proper subset that is itself a diagnosis under the framework. It need not contain the fewest possible components, have the highest probability, or be the cheapest explanation to investigate. These criteria require additional definitions and evidence.

The connection to pattern-oriented analysis is complementary. A recognized artifact pattern can suggest which models or hypotheses deserve examination. A model can then test consistency and identify discriminating observations. The pattern narrows attention; the formal relationships determine what follows under the selected assumptions. Neither operation should silently replace the other.

PART VI

Machine checked mathematics and computational practice

26. Proof assistants kernels and mathematical libraries

The prospect of machine checking changed the scale and representation of mathematical evidence. De Bruijn's AUTOMATH work, described in the proceedings of a 1968 symposium published in 1970, treated mathematical language as something that could be represented for mechanical checking. The aim required explicit definitions and inferential dependencies. [1]

Edinburgh LCF, documented by Gordon, Milner, and Wadsworth in 1979, developed an architecture in which theorem construction was controlled by a small logical core while programmable tactics supported proof development. This separation influenced later systems. A tactic could fail or choose an unhelpful route without being granted unrestricted authority to declare a theorem. [2]

Modern proof assistants differ in their logics, foundations, trusted components, and automation. Lean 4, described by de Moura and Ullrich in 2021, integrates an interactive theorem prover with a programming language. The mathlib project, documented in a 2020 paper, illustrates the collective construction of a reusable formal mathematical library. [66, 67]

A theorem in such a library has dependencies on definitions, lemmas, and sometimes explicit axioms. Acceptance by the checker establishes the claim within that environment. It does not establish that a human reader's informal paraphrase is equivalent to the formal statement, or that a physical interpretation of a formal structure is appropriate.

Proof development creates several diagnostic artifacts. A goal records the current obligation. A context records the available assumptions. An error message can identify an incompatible type or an unresolved reference. A proof term provides evidence for checking. A tactic trace explains part of the construction process. These artifacts describe related but different stages.

Library evolution introduces another kind of failure. A renamed lemma or a changed interface can break a proof script even when the mathematical theorem remains available. A changed definition can instead alter the theorem being proved. Version-aware diagnosis distinguishes a maintenance failure from a mathematical failure before attempting a repair.

A useful preservation record includes the source statement, dependency versions, build instructions, and any admitted axioms or unchecked boundaries. Rechecking a library is strongest when it reconstructs the relevant evidence in a known environment. An old success message by itself does not describe the present state of the artifact.

27. The four color problem and the diagnosis of a proof

The four color problem provides a particularly instructive separation between a theorem and a proposed proof. Kempe published a purported proof in 1879. Heawood's 1890 paper exposed a defect in the argument and established the five-color result. The failure of Kempe's proof did not show that the four-color claim was false. [68, 69]

The relevant method involved interchanging colors within connected configurations, now associated with Kempe chains. The diagnostic issue was whether such changes could always be combined as required. A local recoloring operation could be valid while the overall argument about its interaction with other chains failed. Inspecting one successful rearrangement was insufficient to establish the general construction.

This is a recurring defect mechanism in formal work: operations that are individually permissible can interfere when composed. The missing evidence concerns their relationship. A repair must establish compatibility, choose a different construction, or weaken the conclusion. Simply repeating the local argument in more examples does not fill the gap.

Appel and Haken announced a computer-assisted proof in 1976, followed by detailed publications in 1977. Their approach combined mathematical reductions with extensive computational checking. The resulting proof required readers to consider both the mathematical structure and the trust placed in the computation. [70]

Gonthier and Werner's machine-checked proof, completed in 2005 and explained in Gonthier's 2008 Notices article, brought the argument into Coq. It illustrates a further distinction: using a computer to carry out a large calculation is not the same as having the required logical relationships checked within a proof assistant. [71]

There are therefore several different historical artifacts: the conjecture, Kempe's argument, Heawood's diagnosis, the computer-assisted proof, and the later formal proof. Treating them as versions of one undifferentiated result would conceal the development. Each artifact changes what can be checked and what remains trusted.

For our diagnostic vocabulary, the case also warns against assigning status to a theorem through the fate of one proof attempt. "This step is invalid," "this proof is incomplete," and "this proposition is false" are different findings. Preserving that distinction allows a failed argument to become useful evidence rather than a misleading verdict on the statement itself.

28. Large formal proofs and the decomposition of trust

Large proofs combine many kinds of work: conceptual reductions, finite classification, symbolic calculation, numerical inequalities, and library results. Their scale makes it difficult for any one reader to

inspect every dependency in the same way. Formalization can reorganize this evidence into obligations handled by different checking mechanisms.

The Flyspeck project produced a formal proof of the Kepler conjecture about the densest packing of congruent balls in three-dimensional Euclidean space. The project was completed in 2014, its preprint appeared in 2015, and the published account appeared in Forum of Mathematics, Pi in 2017. The formal proof used HOL Light and Isabelle. [72]

These dates are not competing answers to one question. They identify completion, dissemination, and journal publication. Keeping them separate helps reconstruct how an evidence artifact became available and how its public account developed. It also avoids turning the date of a repository or paper into a claim about when every underlying idea originated.

The machine-checked odd order theorem provides another example. Its formalization was completed in 2012 and described in a 2013 paper by Gonthier and colleagues. The theorem states that every finite group of odd order is solvable. Here “solvable” is a technical group-theoretic property, not a statement that every computational problem about the group is easy. [73]

Large formalizations build reusable infrastructure as well as final theorems. Definitions, algebraic structures, finite arguments, and automation become dependencies for later work. A defect in a shared component can therefore have broad consequences, while a well-organized dependency structure can make rechecking systematic.

The trusted base does not disappear. The proof checker, logical foundations, runtime, compiler, and hardware occupy different places in the chain of trust. Some systems reduce the trusted software core or permit independent checks. Those architectural choices are meaningful, but they should be described specifically rather than summarized as complete freedom from trust.

A diagnostic investigation of a formal proof can therefore ask which boundary has failed. Did a numerical certificate fail to check? Did a lemma use a different convention? Did the formal statement omit a condition in the intended theorem? Did the build environment change? The answer determines whether the repair is mathematical, representational, computational, or archival.

29. Reproducible computation and the history of an artifact

A computational result is produced by a process. Its preservation requires more than the final number or diagram. Inputs, transformations, source code, dependencies, and configuration can all affect what was calculated. Reproducibility makes this process available for renewed examination.

The FAIR principles, published by Wilkinson and colleagues in 2016, concern making data and related resources findable, accessible, interoperable, and reusable. Their application helps others locate and interpret computational artifacts. FAIRness is not a guarantee that a result is correct, and accessibility can involve an authorization procedure rather than unrestricted public availability. [74]

Formal science gives reproducibility several distinct roles. Repeating a numerical experiment checks whether the recorded process produces the same result. Rechecking a proof object tests its acceptance by a checker. Reconstructing a proof from source exercises the build and dependency environment as well. Independently proving the theorem supplies a different kind of confirmation.

Consider a notebook in which cells were executed out of their displayed order. The visible page can suggest a computation that differs from the state actually used. Restarting the environment and running the cells in order may expose the dependency. The defect concerns the correspondence between the displayed narrative and the executed sequence.

Randomness and parallel execution introduce further choices. A seed can help reproduce one pseudorandom sequence, but it does not document every algorithmic or library change that affects the result. A parallel reduction can change floating-point evaluation order. Bitwise equality, numerical agreement within a tolerance, and agreement on a mathematical conclusion are different reproducibility criteria.

Version information is especially useful when a previously accepted artifact stops working. We can compare the input statement, code, dependency graph, and execution or proof trace. The earliest difference may explain the later failure, but temporal precedence alone does not establish causation. A controlled replay or a smaller reproducing case can test the suspected mechanism.

The preserved artifact should therefore include a clear claim about what has been reproduced. A successful rebuild does not validate an incorrect model, while a failed rebuild caused by a missing dependency does not refute a theorem. Reproducibility increases the evidence available for diagnosis when its scope is stated precisely.

30. Machine learning in conjecture and proof search

Machine learning has added methods for selecting premises, proposing proof steps, and searching for mathematical constructions. Its role depends on the surrounding system. A generated candidate can be checked by an exact evaluator or proof assistant; a fluent explanation without such evidence remains a proposed argument.

Polu and Sutskever’s 2020 work applied generative language modeling to automated theorem proving in a Metamath setting. The generated proof steps operated within a formal checking process. The example shows how a learned search component can contribute without making the model’s confidence the criterion of mathematical validity. [75]

LeanDojo, introduced by Yang and colleagues in 2023, provided tools, data, and a retrieval-augmented prover for interaction with Lean. Premise selection is a diagnostic concern as well as a search problem. A relevant theorem may exist but remain unavailable to the search because its statement, name, or required assumptions are not recognized. [76]

FunSearch, first published online in 2023 and in a 2024 Nature volume, paired a language model with program evaluation in mathematical search. AlphaGeometry, published in 2024, combined learned guidance with a symbolic engine for a restricted geometry domain. These systems used different problem representations and checking arrangements. Their results should not be collapsed into one claim that arbitrary generated mathematics has been verified. [77, 78]

The boundary between proposal and evidence remains important. A generated program may pass its evaluator because the evaluator checks only a limited property. A generated proof may establish a formally encoded statement that differs from the intended informal theorem. A benchmark result may reflect the selected tasks and available libraries rather than a general ability to solve all problems in a discipline.

Agentic workflows add a trace of searches, tool calls, revisions, and checks. That trace can explain which candidate was accepted and why. It can also reveal that a later explanation omits failed attempts or attributes a result to evidence that was never obtained. The final narrative should be compared with the actual sequence of formal checks.

The historical change is therefore in the production and selection of candidates and in the scale of search. The logical status of a result still depends on the evidence and the checker’s scope. This gives formal sciences a useful model for AI-assisted diagnostics: allow broad proposal generation, and preserve explicit boundaries where claims become established results.

PART VII

Diagnostic cases structural relationships and patterns

31. Vacuity missing cases and misleading success

Formal analysis can report success while leaving the intended question unanswered. This happens when the statement checked is too weak, its assumptions are inconsistent, or the relevant case has been excluded. The checker may be working correctly. The problem lies in the relationship between the formal obligation and the intended requirement.

Consider a specification saying that every request eventually completes, together with an assumption that prevents all requests. The implication may hold vacuously. No counterexample is available because the model never reaches the situation that the property was meant to govern. A reachability check for a request supplies evidence that the liveness requirement alone does not provide.

In a second example, a function is specified to return a nonnegative value but is intended to compute a square root. A constant zero implementation satisfies the stated range condition. The missing relationship concerns the input and the square of the output. Strengthening the specification changes the obligation from an incidental property to a more informative account of the function.

A third example concerns inconsistency. In classical logic, contradictory assumptions allow every conclusion to be derived. If a proof environment has admitted inconsistent axioms, the successful derivation of a desired result cannot provide the intended assurance. Inspecting the assumptions and demonstrating that relevant models exist can expose a problem that ordinary proof replay would preserve.

Missing boundary cases create less dramatic but common failures. An algorithm can be correct for nonempty collections while failing on an empty one. A geometric construction can require distinct points. A division can require a nonzero denominator. The correct response is to decide whether the intended domain excludes the case or whether the implementation or proof must handle it.

These examples are constructions for explanation, not historical incident reports. They draw together distinctions that became explicit through program logic, satisfiability analysis, and model checking. The historical methods provide different ways to investigate them: assertions expose missing relationships, satisfiability checks examine compatible assumptions, and state exploration examines whether the intended scenario can occur. [54, 24, 57]

We should therefore examine a success artifact with the same care as a failure artifact. What was proved? Under which assumptions? Does the relevant situation exist in the model? Which cases were excluded? These questions do not weaken a formal result. They establish its correspondence with the problem that motivated the analysis.

32. Categories composition and structural consistency

Many diagnostic problems concern the correspondence between representations. A mathematical object is encoded as a formula, translated into a solver language, and interpreted after computation. Even if each local step has a plausible description, the composition can fail to preserve the relationship that matters.

Eilenberg and Mac Lane's 1945 General Theory of Natural Equivalences introduced categories, functors, and natural transformations in a systematic form. These notions describe objects together with structure-preserving mappings and relationships among those mappings. Their original mathematical development provides a language that can also be used to examine transformations between formal representations. [79]

A commuting diagram expresses agreement between specified routes of composition. For diagnosis, the useful question is concrete: does translating and then transforming give the same result, in the required sense, as transforming and then translating? Equality is one possible criterion. In other settings, an equivalence relation or a justified approximation is the appropriate criterion.

Consider an optimization model converted from one system of units to another. The conversion of constraints and the conversion of a candidate solution should correspond. If the objective is converted but one constraint is left in the original units, the two routes disagree. A structural description helps locate the inconsistent mapping; it does not determine the correct conversion without the meanings of the quantities.

Rutten's 2000 account of universal coalgebra developed a general theory of state-based systems and behavior. Relations such as bisimulation provide structured ways to compare systems under a chosen behavioral interpretation. Two systems can be equivalent for one observation scheme while differing internally or under a richer set of observations. [80]

This connects composition with observability without making them identical. A mapping can preserve the outputs relevant to one analysis while discarding the information required for another. An abstraction that is appropriate for a safety proof may be unsuitable for explaining a performance difference. The property and the observation scheme determine which relationships must survive.

Our application of categorical language to diagnostics is an analytical interpretation. A collection of boxes and arrows is not automatically a category, and a visual similarity does not establish a functor. We need to define the objects, mappings, identities, composition, and preserved structure before claiming the corresponding mathematical result. This keeps a useful organizing vocabulary connected to explicit obligations.

33. Pattern oriented diagnosis of formal artifacts

In our pattern-oriented software diagnostics, an analysis pattern describes a recurring structure in an artifact or in relationships among artifacts. It helps us recognize a situation and choose further analysis. A defect mechanism explains how a particular problem was produced. The two levels are related, but recognizing the pattern does not by itself establish the mechanism. [81]

Formal work provides additional artifacts to which this distinction can be applied. A proof-state history, an optimizer log, a sampler trace, and a sequence of model-checker refinements can all preserve activity. Their messages require their own semantics. A failed proof obligation and an excessive residual are both noteworthy events, but they do not carry the same logical force.

Several patterns from our trace analysis reference offer useful starting points. Message Context concerns a set of surrounding messages related to a selected message. Message Invariant concerns the invariant part of message text, distinguished from its variable data; it is not the same concept as a proved loop invariant. Missing Message raises a question about expected evidence and its collection. Meta Trace concerns the evolution of traces and logs as software changes. These are existing pattern names; their applications below extend the analysis setting. [82]

For example, messages recording a model version and solver configuration can provide Message Context for a later “unsatisfiable” message. The encoded assumptions remain necessary supporting information. A missing “proof complete” message can mean an interrupted run, a capture failure, or an unfinished proof; its absence alone does not select among them. Comparing proof-state logs across versions of a proof assistant or its libraries offers an application of Meta Trace. A log of one rechecking run, by itself, does not establish that evolutionary comparison.

Adjoint Thread of Activity provides a way to group messages by a relationship beyond an operating-system thread identifier. In a proof workflow, a grouping might follow one theorem obligation across several tactics and tools. In an optimization workflow, it might follow one constraint through parsing, presolve, conflict analysis, and the final explanation. The grouping criterion must be stated so that unrelated messages are not combined merely because they share a convenient label. [82]

Our 4WS questions from “Five Golden Rules” can also be adapted: What, When, Where, Why, and Supporting information. What discrepancy was observed? When, or at which stage, did it appear? Where did the artifact originate? Why was it collected or requested? What supporting information is needed to interpret it? In formal sciences, that information includes domains, axioms, definitions, bounds, tolerances, and dependency versions. [81]

These applications form a bridge between methods. They do not make statistical evidence deductive, or convert a trace pattern into a proof. They help arrange the evidence so that the appropriate mathematical, statistical, or computational method can be applied at the correct level.

34. Review institutions and the preservation of mathematical evidence

Formal results are produced and maintained by communities. Journals, correspondence, monographs, repositories, and libraries determine how arguments become available for inspection and correction. Their practices influence the visibility of defects even when the logical validity of a theorem does not depend on a vote.

The four-color history shows several forms of scrutiny: a published argument, a later diagnosis of its defect, a computer-assisted proof, and a machine-checked reconstruction. The large formalizations of the odd order theorem and the Kepler conjecture show how shared libraries and coordinated work can distribute the construction of evidence. These examples describe different arrangements of human and mechanical checking. [71, 73, 72]

Peer review and machine checking answer overlapping but distinct questions. A reviewer may identify an unclear definition, an inappropriate application, or an unsupported interpretation. A checker can examine derivations at a level of detail that is impractical for routine manual review. Either process can miss what lies outside its scope.

An erratum is also a diagnostic artifact. It records a difference between an earlier claim or argument and its correction. A useful erratum identifies what changes, which results depend on the change, and what remains valid. Replacing a file without preserving that relationship can leave readers with incompatible versions and no explanation of their connection.

Collective formal libraries make dependency management part of mathematical maintenance. A change to a central definition can require many proofs to be rebuilt. A local renaming can require many scripts to be repaired without changing their mathematical content. The social organization and architecture described by the mathlib community address parts of this continuing work. [67]

Preservation must also include the environment needed to interpret an artifact. An unavailable compiler or undocumented proof format can make a formally checkable result difficult to recheck. A readable mathematical account remains valuable even when a machine artifact is retained, because it explains the result’s intended meaning and the organization of its evidence.

The history of formal diagnostics therefore includes correction practices and evidence maintenance alongside major theorems. They determine whether a later analyst can distinguish an established result, a repaired argument, an obsolete implementation, and an unrepeatable claim.

35. Continuing developments and the limits of automation

The developments considered here move some diagnostic work from informal inspection into explicit, repeatable procedures. More assumptions can be checked. Larger state spaces can be explored. Proof

search can be guided by learned methods. Numerical results can be accompanied by bounds or certificates. These changes also produce new layers whose relationships require investigation.

One continuing direction is the composition of evidence. A verified transformation can connect a high-level statement to a solver input. A solver can return a certificate. A separate checker can validate the certificate. The overall conclusion depends on the correspondence between all these stages, including the original interpretation of the problem.

Another direction concerns explanations. A large proof object can establish validity without making the central idea easy to understand. A minimal conflict can identify incompatible assumptions without explaining which assumption should change. A diagnostic system benefits from both checkable evidence and an account of what that evidence means for the intended problem.

Learned search and agentic workflows increase the importance of preserving unsuccessful attempts and tool outcomes. A generated explanation can omit an “unknown” result or present a proposed step as checked. Recording the actual proof and computation events makes these discrepancies available for analysis. The evidence should remain identifiable independently of the fluency of the final narrative. [75, 76]

The limitations established earlier remain in force. Finite testing does not prove an unrestricted universal claim. A theorem about a model does not establish that the model represents an empirical system. An identifiable parameter in an ideal experiment can be poorly determined by actual data. A mathematically observable system can be numerically difficult to reconstruct.

Our practical response is to keep the diagnostic question explicit. We select the artifact that can address it, preserve the assumptions required to interpret that artifact, and state the result at the level supported by the evidence. When alternatives remain indistinguishable, we describe the missing distinction instead of replacing it with a more confident explanation.

Formal sciences contribute more than tools for obtaining answers. They provide methods for examining the answers, the questions, and the boundaries between them. Their continuing history is also a history of making those boundaries available for analysis.

Conclusion

The history of diagnostics in formal sciences follows changes in what can be examined. A recorded procedure makes a calculation repeatable. A proof exposes dependencies among statements. An axiom system makes foundational commitments explicit. A residual shows a discrepancy under a fitted model. A state-space description identifies what outputs can distinguish. A certificate separates an expensive search from a checkable result.

These artifacts do not form one scale on which every later method replaces an earlier one. Substitution into an original equation remains useful beside symbolic computation. A small counterexample can be more informative than a large search log. Human interpretation remains necessary when the formal statement and its intended meaning must be connected. Different methods preserve different relationships.

Several recurring distinctions organize the history. A statement can be false while one proof attempt is merely incomplete. A model can be mathematically consistent while unsuitable for its application. A numerical algorithm can be stable while the problem is ill-conditioned. A solver can answer its input correctly while the encoding asks the wrong question. A machine-checked derivation can depend on an assumption that the reader did not intend to accept.

For pattern-oriented diagnostics, these distinctions identify where to look. We examine context, invariants, missing evidence, dependency changes, and the correspondence between representations. We use patterns to organize the investigation and appropriate formal methods to establish what follows. The result is stronger when its assumptions and remaining alternatives are visible.

A diagnosis is complete for its purpose when the evidence supports the proposed explanation and the correction has been checked against the original problem. In some cases, the result is instead a precise statement of what cannot yet be distinguished. Formal sciences provide methods for both outcomes. Their contribution is the ability to say more clearly what has been established, how it was established, and where another observation, construction, or argument is still needed.

Appendix A. Selected chronology

Dates below identify the particular development or artifact cited. Ancient dates are approximate where indicated. Completion, announcement, translation, and publication are distinguished when they differ.

Date	Development	Diagnostic significance and source
Fourth century BCE	Aristotle's Prior Analytics	Forms of inference are distinguished from the truth of their premises. [13]
Around 300 BCE	Composition of Euclid's Elements	Connected demonstrations make dependencies available for inspection. [7]
263 CE	Liu Hui's commentary on the Nine Chapters	Explanations connect procedures with the relationships that justify them. [4]
Seventh century onward	Brahmagupta's work and later Indian algebraic traditions	Rules and worked problems preserve calculational structure; represented here through Colebrooke's 1817 translation. [5]
Ninth century	Al-Khwārizmī's algebraic treatise	Organized procedures and geometrical justification for equation solving. [6]
888	Copying of MS D'Orville 301	A dated surviving artifact of the transmission of the Elements. [7]
1763	Publication of Bayes's essay	An inverse probability problem links evidence and uncertain inference. [25]
1817	Colebrooke's translation	Makes selected Indian arithmetic and algebraic texts available in English. [5]
1821	Cauchy's Cours d'analyse	Systematic treatment of limits, continuity, and convergence. [9]
1831	Rosen's translation of al-Khwārizmī	Preserves an English account of the earlier algebraic text. [6]
1854	Boole's Laws of Thought	Algebraic representation supports examination of logical relationships. [14]
1879	Frege's Begriffsschrift	Formal representation of inference and quantification. [15]
1879	Kempe's proposed four-color proof	A later-diagnosed proof defect becomes a major case of mathematical correction. [68]
1890	Heawood's Map-Colour Theorem	Diagnosis of Kempe's argument and the five-color result. [69]
1899	Hilbert's Grundlagen der Geometrie	Explicit axioms support consistency and independence investigations. [8]
1900 and 1902	Hilbert's address and English publication	Foundational problems become an explicit research program. [19]
1903	Russell's Principles of Mathematics	Public discussion of a foundational contradiction. [16]
1908	Zermelo's axiomatization	Set-formation assumptions become objects of explicit restriction. [17]
1922	Fisher's foundations paper	Properties of estimation procedures receive systematic treatment. [27]

Date	Development	Diagnostic significance and source
1931	Gödel's incompleteness paper	Limits of provability within sufficiently strong effective systems. [20]
1933	Kolmogorov's probability axioms	Measure-theoretic framework for probability. [26]
1933	Neyman and Pearson on hypothesis testing	Error probabilities and alternatives structure testing procedures. [30]
1935	Gentzen on logical deduction	Derivations and inference rules become structured objects of analysis. [18]
1936	Church's unsolvability paper	Limits of effective decision procedures. [21]
1936–1937	Turing's computability paper	Explicit computation model and undecidability results. [22]
1945	Eilenberg and Mac Lane	Categories, functors, and natural transformations describe structural relationships. [79]
1948	Shannon's communication theory	Information, coding, and channel constraints receive a unified mathematical treatment. [38]
1950	Hamming's coding paper	Redundancy supports specified error detection and correction. [39]
1950	Nash's equilibrium note	Existence of mixed-strategy equilibria for finite games. [49]
1951	Kuhn and Tucker's symposium paper	Constraint and multiplier relationships support nonlinear optimization analysis. [46]
1960	Kalman's filtering paper	Model-based recursive state estimation. [42]
1960 and 1961	Kalman's IFAC contribution and proceedings	Observability, controllability, and their duality are introduced at the 1960 congress; proceedings follow in 1961. [40]
1960	Davis and Putnam's procedure	Algorithmic treatment of logical satisfiability and deduction. [60]
1963	Kalman's systems description	The earlier observability and controllability concepts receive a systematic realization treatment. [41]
1963	Wilkinson's rounding-error book	Computed results are related to exact and nearby mathematical problems. [35]
1963	Tikhonov's regularization paper	Additional structure stabilizes selected ill-posed problems. [45]
1963–1964	Lakatos's Proofs and Refutations serial	A philosophical dialogue examines conjectures, proofs, counterexamples, and revised definitions. [10]
1965	Robinson's resolution principle	Uniform inference supports automated deduction. [61]
1967	Floyd on program meaning	Assertions connect program structure with correctness obligations. [53]
1969	Hoare's axiomatic basis	Preconditions and postconditions organize program reasoning. [54]
1970	Publication of de Bruijn's AUTOMATH account	Formal mathematical language supports mechanical checking. [1]
1970	Bellman and Åström	Structural identifiability distinguishes parameter inference from fitting. [44]

Date	Development	Diagnostic significance and source
1971	Scott and Strachey	Mathematical semantics clarifies the meaning of program constructions. [51]
1971	Cook’s complexity result	Resource analysis separates efficient checking from search. [23]
1976	Appel and Haken’s announcement	A computer-assisted proof of the four-color theorem is announced; detailed papers follow in 1977. [70]
1976	Lakatos’s Proofs and Refutations book	The expanded book develops the earlier dialogue and further cases of mathematical discovery. [10]
1976	Box on science and statistics	Model criticism is connected with model construction. [29]
1976	Willsky’s failure-detection survey	Dynamical models organize residual-based fault investigation. [43]
1977	Cook’s influence measure	Observation influence is distinguished from residual magnitude. [28]
1977	Cousot and Cousot	Sound abstraction provides a foundation for static analysis. [55]
1977	Pnueli’s temporal logic paper	Program properties are expressed over behavior in time. [56]
1978	Milner on type polymorphism	Type constraints provide a formal basis for checking reusable programs. [52]
1979	Edinburgh LCF	Programmable proof construction is organized around a logical core. [2]
1981–1982	Early model checking	Temporal properties are checked over transition systems. [57, 58]
1987	Reiter’s diagnosis theory	Conflicts and candidate diagnoses are related formally. [65]
1991	Chinneck and Dravnieks	Incompatible constraint subsets support formulation diagnosis. [48]
1995	Benjamini and Hochberg	False discovery rate becomes a target of multiple-testing control. [31]
1995	Pearl’s causal diagrams paper	Intervention and identification are examined under graphical assumptions. [50]
1996	Gelman Meng and Stern	Posterior predictive discrepancies support model checking. [32]
2000	Counterexample-guided abstraction refinement	Spurious counterexamples direct refinement. [59]
2000	QuickCheck	Executable properties generate tests. [11]
2000	Rutten’s universal coalgebra account	State and behavior are organized through general structural relationships. [80]
2002	Alloy and delta debugging publications	Bounded model search and systematic reduction provide compact failure evidence. [24, 12]
2005 and 2008	Gonthier’s four-color formalization and exposition	Machine-checked proof and its published explanation are distinct artifacts. [71]

Date	Development	Diagnostic significance and source
2006	SAT modulo theories framework	Boolean search is combined with reasoning in interpreted theories. [62]
2012–2013	Odd order theorem formalization and paper	Large algebraic libraries support a machine-checked proof. [73]
2014	DRAT-trim	Expressive clausal proof evidence can be checked and trimmed. [63]
2014–2017	Flyspeck completion preprint and publication	A large geometric proof is formalized across proof assistants. [72]
2016	Boolean Pythagorean triples result	SAT search produces a checked mathematical refutation certificate. [64]
2016	FAIR principles	Computational artifacts gain explicit reuse and provenance requirements. [74]
2018 and 2020	Simulation-based calibration preprint and revision	Simulated inference experiments examine computational calibration. [34]
2020	mathlib paper and generative theorem proving	Shared formal libraries and learned proof search develop together. [67, 75]
2021	Lean 4 paper and improved MCMC diagnostics	Distinct advances in proof environments and computational checking. [66, 33]
2023	LeanDojo	Reproducible tooling supports retrieval-assisted theorem proving. [76]
2023–2024	FunSearch online and volume publication	Learned proposal generation is coupled to program evaluation. [77]
2024	AlphaGeometry	Learned guidance and symbolic deduction are combined in a geometry domain. [78]

Appendix B. Glossary

Abstraction. A representation that preserves selected relationships while omitting or combining other details. Its adequacy depends on the property being investigated.

Abstract interpretation. A framework for sound approximation of program semantics through relationships between concrete and abstract domains.

Axiom. A statement accepted as a starting assumption within a formal theory. The consequences of accepting it depend on the surrounding logic and other assumptions.

Backward error. A measure of the change to the input problem needed to make a computed answer an exact solution of the changed problem.

Bisimulation. A relation that matches specified observations and transitions between state-based systems according to the chosen definition.

Bounded analysis. Investigation restricted to a stated number of objects, trace length, numerical range, or other finite scope.

Calibration. Agreement between probabilistic claims and their expected behavior under a specified repeated-use or simulation interpretation.

Category. A mathematical structure consisting of objects and morphisms with associative composition and identity morphisms.

Certificate. Evidence that a result can be checked against a specified verification procedure. Its force depends on the encoding and checking rules.

Commuting diagram. A diagram in which specified composite mappings agree. It expresses an equality obligation rather than a decorative arrangement.

Completeness. A property whose meaning depends on context, such as every semantically valid formula being derivable, or a decision procedure finding every required kind of answer.

Conditioning. Sensitivity of a mathematical problem’s answer to changes in its input, under specified measures of change.

Consistency. In the usual classical deductive setting, the absence of a derivable contradiction. Other settings require their own precise definition.

Constraint. A condition restricting admissible objects, assignments, states, or solutions.

Counterexample. An instance within the stated domain that satisfies the assumptions of a claim and violates its conclusion.

Counterexample-guided abstraction refinement. An iterative method that analyzes abstract counterexamples and refines an abstraction when they do not correspond to concrete behavior.

Defect mechanism. A relationship or process explaining how a particular discrepancy or failure was produced.

Denotational semantics. An account that assigns mathematical meanings to language constructions and relates compound meanings to their parts.

Diagnosis. An evidence-based account of a problem and the explanations compatible with it; in a formal framework, the term can have a narrower defined meaning.

Duality gap. The difference between appropriate primal and dual objective values. Its interpretation requires feasibility and the relevant optimization framework.

Effective procedure. A procedure expressible within a specified formal account of algorithmic computation.

Encoding. A representation of a problem in another language or structure. Its correctness concerns preservation of the intended relationship.

Evidence artifact. A retained object used in an investigation, such as a proof, trace, assignment, residual plot, certificate, or model description.

False discovery rate. The expected proportion of false rejections among all rejections, with the proportion defined as zero when none are made.

Feasibility. Satisfaction of the constraints defining an admissible solution.

Formalization. Representation of statements, definitions, and arguments in a language with explicit syntactic and semantic rules.

Forward error. Difference between a computed answer and the desired exact answer, under a specified measure.

Functor. A mapping between categories that preserves identities and composition.

Identifiability. The ability to distinguish parameter values through the observations or experiments permitted by a model.

Ill-posed problem. A problem failing a relevant requirement of existence, uniqueness, or stability in the chosen formulation.

Independence of an axiom or statement. Non-derivability from the specified remaining assumptions; stronger two-sided independence also concerns non-derivability of the negation.

Influence. The effect an observation has on a fitted statistical result, under the chosen measure or perturbation.

Invariant. A property preserved under specified transformations or transitions. Observed repetition alone does not prove invariance.

Irreducible infeasible set. An infeasible constraint subset that becomes feasible when any member is removed. It need not be the smallest infeasible subset.

Kernel. The core mechanism responsible for checking or constructing trusted logical judgments in a proof-system architecture.

Likelihood. A function of model parameters obtained by evaluating the probability mass or density of observed data under those parameters.

Liveness. A temporal property requiring a specified desirable event or progress to occur, under the stated execution assumptions.

Meta Trace. In our pattern vocabulary, a narrative about a trace or log and its evolution as the software producing it changes.

Model. A mathematical structure satisfying a theory, or a representation used to investigate a system. The intended sense must be stated.

Model checking. Algorithmic examination of whether a transition-system representation satisfies a specified formal property.

Natural transformation. A structure relating functors through component morphisms that satisfy a compatibility condition.

Numerical stability. A property of an algorithm concerning how its computation handles or amplifies errors, interpreted within a specified error analysis.

Observability. In systems theory, distinguishability of internal states through the available outputs under a specified model and experiment.

Operational semantics. An account of program meaning in terms of execution rules, transitions, or evaluations.

Partial correctness. The guarantee that if a program begins in an allowed state and terminates, its result satisfies the stated postcondition.

Posterior predictive check. Comparison of selected features of observed data with corresponding features generated through a fitted Bayesian model.

Proof object. A representation of deductive evidence that can be examined by a checker using specified rules.

Proof script. Instructions used to construct or reconstruct a proof. Its operational behavior can depend on tool and library versions.

Provenance. The recorded origin and transformation history of an artifact.

Regularization. Addition of structure, penalties, or restrictions to stabilize or select solutions of an inverse or estimation problem.

Residual. A discrepancy between an observation and a fitted or predicted quantity under a specified model.

Safety. A temporal property that excludes a specified undesirable event or state; a violation can be witnessed by a finite bad prefix in the standard setting.

Satisfiability. Existence of an interpretation or assignment that makes the specified constraints or formulas true.

Simulation-based calibration. Examination of an inference procedure through repeated parameter and data simulation under a specified generative model.

Soundness. Preservation of the relevant notion of validity or truth by inference rules or an analysis method, relative to its semantics.

Spurious counterexample. An abstract failure artifact that does not correspond to a concrete behavior under the intended relationship.

Structural identifiability. Parameter distinguishability under idealized observations and the structural assumptions of the model.

Syndrome. A pattern of violated code constraints used to detect or identify errors under a specified coding model.

Total correctness. Partial correctness together with termination under the stated assumptions.

Trusted base. The components and assumptions relied on without being checked by the particular verification process under discussion.

Undecidability. Nonexistence of a uniform algorithm deciding every instance of a specified decision problem within the adopted computational framework.

Unsatisfiable core. A subset of constraints that is itself unsatisfiable. Minimality is an additional property.

Vacuity. Satisfaction of a property without exercising the intended substantive condition, such as an implication whose antecedent never occurs.

Verification. Checking conformance to a specification or formal property. Its scope is determined by the property and the model being checked.

Witness. An explicit object supporting an existential or counterexample claim, subject to checking that it has the required properties.

Appendix C. Evidence taxonomy and blind spots

An artifact establishes a result only through an interpretation. The table distinguishes common artifact types by their normal use and by what they do not establish on their own. Particular tools can provide additional guarantees; those guarantees should be recorded with the artifact.

Artifact	What it can support	Principal blind spot
Written calculation	Repeated examination of operations and substitutions	Correct arithmetic can use the wrong problem data or domain
Informal proof	Human inspection of an argument and its dependencies	Implicit assumptions and omitted cases can remain unnoticed
Machine-checked proof	Derivability of an encoded statement within the checked environment	Translation, admitted axioms, and trusted components remain relevant
Counterexample	Refutation of a universal claim when the instance is admissible and correctly checked	A spurious or out-of-domain instance does not refute the intended claim
Satisfying assignment	Compatibility of the encoded constraints	Missing constraints can make the encoded problem too weak
Unsatisfiability certificate	Nonexistence of a solution to the checked encoding	The encoding may exclude intended solutions
Unsatisfiable core	A conflicting subset of the constraints	It need not be minimal or identify which requirement should change
Model-checker trace	A behavior permitted by the model that violates the property	It may be abstract, spurious, or absent from actual execution
Residual plot	Structure in model-data discrepancies	The shape does not uniquely identify a mechanism
Statistical test result	A discrepancy or decision under a specified model and procedure	It does not directly establish causality or hypothesis truth
Posterior sample	Approximation to a conditional distribution under computational assumptions	Poor exploration or model mismatch may persist
Calibration experiment	Behavior of a procedure across the specified simulated cases	Shared simulator defects or irrelevant test distributions can be missed
Error bound or interval	A quantified relationship to an exact result under stated assumptions	Input uncertainty or dependency can limit its usefulness
Observability analysis	State distinguishability under the selected dynamics and outputs	Exact observability does not guarantee accurate reconstruction from noisy data
Identifiability analysis	Whether parameters can be distinguished in the proposed setup	Ideal distinguishability does not ensure precise practical estimation
Optimization certificate	Feasibility bounds or optimality under the formulation and arithmetic assumptions	A correct optimum can optimize the wrong objective
Build or replay log	Evidence of the process used to reconstruct an artifact	A repeated process can reproduce the same defect
AI-generated argument	A candidate explanation or route for further checking	Fluency and confidence do not establish validity

C1 Questions to carry with an artifact

First, identify the claim. Is it about a proposition, an encoded model, a program, a numerical answer, or an empirical application? Next, identify the assumptions that give the artifact its meaning. These can include axioms, domains, probability models, observation maps, bounds, fairness conditions, precision, and dependency versions.

Then identify the checking operation. A human review, kernel check, finite search, statistical test, and numerical enclosure establish different kinds of result. Record the actual outcome, including an "unknown" result, timeout, incomplete proof, or failed replay. These outcomes should remain visible in later summaries.

Finally, identify what alternatives remain. Can several states produce the same outputs? Can several models fit the observations? Does the failure belong to the statement, its representation, the checking method, or the surrounding environment? A next step is useful when it supplies evidence that distinguishes a relevant alternative.

C2 Combining evidence without losing its meaning

Several artifacts can support one investigation without becoming interchangeable. A numerical experiment can suggest a conjecture. A counterexample search can expose a missing condition. A proof can establish the revised statement. A formalization can make its dependencies mechanically checkable. A build log can preserve how that evidence was reconstructed.

The combined record is strongest when these relationships are explicit. We should be able to trace a conclusion to the artifact that supports it and to the assumptions required for that support. A diagram, table, or generated explanation can help navigate the record, but it should not replace the underlying proof, data, or certificate.

This is also the boundary of a pattern-oriented interpretation. A recurring arrangement can guide attention across several artifacts. Its logical or statistical force still comes from the formal relationships in the particular case. Preserving that distinction allows methods from different formal sciences to cooperate in one diagnosis.

References

References are numbered in order of first citation. Original publication dates are distinguished from translations, reprints, and later accounts. URLs are printed explicitly for use with a paper copy.

[1] de Bruijn, N. G. The Mathematical Language AUTOMATH, Its Usage, and Some of Its Extensions. Symposium on Automatic Demonstration, LNM 125. Springer, 1970, 29–61; symposium held in 1968.
https://doi.org/10.1007/BFb0060623
https://research.tue.nl/en/publications/the-mathematical-language-automath-its-usage-and-some-of-its-exte/

[2] Gordon, M. J., R. Milner, and C. P. Wadsworth. Edinburgh LCF: A Mechanized Logic of Computation. LNCS 78. Springer, 1979.
https://doi.org/10.1007/3-540-09724-4

[3] Higham, N. J. Accuracy and Stability of Numerical Algorithms. Second edition. SIAM, 2002.
https://doi.org/10.1137/1.9780898718027

[4] Shen, K., J. N. Crossley, and A. W.-C. Lun. The Nine Chapters on the Mathematical Art: Companion and Commentary. Oxford University Press, 1999. Translation of the ancient text with its commentarial tradition.
https://academic.oup.com/book/52966

[5] Brahmagupta and Bhāskara. Algebra, with Arithmetic and Mensuration, from the Sanscrit of Brahmegupta and Bhascara. Translated by H. T. Colebrooke. John Murray, 1817.
https://books.google.com/books?id=zqMIAAAAQAAJ

[6] Al-Khwārizmī. The Algebra of Mohammed ben Musa. Edited and translated by F. Rosen. Oriental Translation Fund, 1831. Translation of the ninth-century algebraic treatise.
https://archive.org/details/algebraofmohamme00khuwuoft

[7] Euclid. Elements. Composed around 300 BCE. Greek manuscript MS D’Orville 301, copied in 888 CE; digital edition, Clay Mathematics Institute and Bodleian Library.
https://www.claymath.org/library/historical/euclid/

[8] Hilbert, D. The Foundations of Geometry. Translated by E. J. Townsend, 1902, from Grundlagen der Geometrie, 1899.
https://www.gutenberg.org/ebooks/17384

[9] Cauchy, A.-L. Cours d’analyse, 1821. English edition: R. E. Bradley and C. E. Sandifer, Cauchy’s Cours d’analyse: An Annotated Translation. Springer, 2009.
https://doi.org/10.1007/978-1-4419-0549-9

[10] Lakatos, I. Proofs and Refutations: The Logic of Mathematical Discovery. Edited by J. Worrall and E. Zahar. Cambridge University Press, 1976. Editors’ Preface, pp. ix–xi, describes the four-part serial in the British Journal for the Philosophy of Science 14 (1963–1964).
https://doi.org/10.1017/CBO9781139171472

[11] Claessen, K., and J. Hughes. QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs. Proceedings of ICFP 2000, 268–279.
https://doi.org/10.1145/351240.351266

[12] Zeller, A., and R. Hildebrandt. Simplifying and Isolating Failure-Inducing Input. IEEE Transactions on Software Engineering 28 (2002), 183–200.
https://doi.org/10.1109/32.988498

[13] Aristotle. Prior Analytics. Fourth century BCE. Translated by A. J. Jenkinson. Internet Classics Archive.
https://classics.mit.edu/Aristotle/prior.html

[14] Boole, G. An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities. Walton and Maberly, 1854.
https://www.gutenberg.org/ebooks/15114

[15] Frege, G. Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Louis Nebert, 1879.
https://zenodo.org/records/6330677

[16] Russell, B. The Principles of Mathematics. Volume I. Cambridge University Press, 1903.
https://archive.org/details/principlesmathe00russgoog

[17] Zermelo, E. Untersuchungen über die Grundlagen der Mengenlehre. I. Mathematische Annalen 65 (1908), 261–281.
https://doi.org/10.1007/BF01449999

[18] Gentzen, G. Untersuchungen über das logische Schließen. I. Mathematische Zeitschrift 39 (1935), 176–210.
https://doi.org/10.1007/BF01201353

[19] Hilbert, D. Mathematical Problems. Translated by Mary Winston Newson from the 1900 Paris address. Bulletin of the American Mathematical Society 8 (1902), 437–479.
https://aleph0.clarku.edu/~djoyce/hilbert/problems.html

[20] Gödel, K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik 38 (1931), 173–198.
https://doi.org/10.1007/BF01700692

[21] Church, A. An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics 58 (1936), 345–363.
https://doi.org/10.2307/2371045

[22] Turing, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, series 2, 42 (1936–1937), 230–265. The publisher catalogs the paper under 1937.
https://doi.org/10.1112/plms/s2-42.1.230

[23] Cook, S. A. The Complexity of Theorem-Proving Procedures. Proceedings of STOC 1971, 151–158.
https://doi.org/10.1145/800157.805047

[24] Jackson, D. Alloy: A Lightweight Object Modelling Notation. ACM Transactions on Software Engineering and Methodology 11 (2002), 256–290. The second link is the author’s preprint.
https://doi.org/10.1145/505145.505149
https://groups.csail.mit.edu/sdg/pubs/2002/alloy-journal.pdf

[25] Bayes, T. An Essay towards Solving a Problem in the Doctrine of Chances. Communicated by R. Price. Philosophical Transactions 53 (1763), 370–418.
https://doi.org/10.1098/rstl.1763.0053

[26] Kolmogorov, A. N. Grundbegriffe der Wahrscheinlichkeitsrechnung. Springer, 1933. English translation edited by N. Morrison: Foundations of the Theory of Probability, Chelsea, 1950; second English edition, 1956.

[27] Fisher, R. A. On the Mathematical Foundations of Theoretical Statistics. Philosophical Transactions of the Royal Society A 222 (1922), 309–368.
https://digital.library.adelaide.edu.au/items/36b96da4-e256-4e94-9720-eef372935c3a

[28] Cook, R. D. Detection of Influential Observation in Linear Regression. Technometrics 19 (1977), 15–18.
https://doi.org/10.1080/00401706.1977.10489493

[29] Box, G. E. P. Science and Statistics. Journal of the American Statistical Association 71 (1976), 791–799.
https://doi.org/10.1080/01621459.1976.10480949

[30] Neyman, J., and E. S. Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. Philosophical Transactions of the Royal Society A 231 (1933), 289–337.
https://doi.org/10.1098/rsta.1933.0009

[31] Benjamini, Y., and Y. Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society B 57 (1995), 289–300.
https://doi.org/10.1111/j.2517-6161.1995.tb02031.x

[32] Gelman, A., X.-L. Meng, and H. Stern. Posterior Predictive Assessment of Model Fitness via Realized Discrepancies. Statistica Sinica 6 (1996), 733–807, including discussion and rejoinder.
https://www3.stat.sinica.edu.tw/statistica/j6n4/j6n41/j6n41.htm

[33] Vehtari, A., A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. Rank-Normalization, Folding, and Localization: An Improved R-hat for Assessing Convergence of MCMC, with discussion. Bayesian Analysis 16 (2021), 667–718.
https://doi.org/10.1214/20-BA1221

[34] Talts, S., M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman. Validating Bayesian Inference Algorithms with Simulation-Based Calibration. arXiv:1804.06788, first submitted 2018; revised 2020.
https://arxiv.org/abs/1804.06788

[35] Wilkinson, J. H. Rounding Errors in Algebraic Processes. Prentice-Hall (USA) and HMSO (UK), 1963. Reissued in the SIAM Classics series, 2023.
https://doi.org/10.1137/1.9781611977523

[36] Goldberg, D. What Every Computer Scientist Should Know about Floating-Point Arithmetic. ACM Computing Surveys 23 (1991), 5–48.
https://doi.org/10.1145/103162.103163

[37] Moore, R. E., R. B. Kearfott, and M. J. Cloud. Introduction to Interval Analysis. SIAM, 2009.
https://doi.org/10.1137/1.9780898717716

[38] Shannon, C. E. A Mathematical Theory of Communication. Bell System Technical Journal 27 (1948), 379–423 and 623–656. The links identify the two installments.
https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
https://doi.org/10.1002/j.1538-7305.1948.tb00917.x

[39] Hamming, R. W. Error Detecting and Error Correcting Codes. Bell System Technical Journal 29 (1950), 147–160.
https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[40] Kalman, R. E. On the General Theory of Control Systems. First IFAC Congress, Moscow, 1960; proceedings published as Automatic and Remote Control, vol. 1, Butterworths, 1961. Digital record: IFAC Proceedings Volumes 1 (1960), 491–502.
https://doi.org/10.1016/S1474-6670(17)70094-8

[41] Kalman, R. E. Mathematical Description of Linear Dynamical Systems. Journal of the Society for Industrial and Applied Mathematics, Series A: Control 1 (1963), 152–192.
https://doi.org/10.1137/0301010

[42] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering 82 (1960), 35–45.
https://doi.org/10.1115/1.3662552

[43] Willsky, A. S. A Survey of Design Methods for Failure Detection in Dynamic Systems. Automatica 12 (1976), 601–611.
https://doi.org/10.1016/0005-1098(76)90041-8

[44] Bellman, R., and K. J. Åström. On Structural Identifiability. Mathematical Biosciences 7 (1970), 329–339.
https://doi.org/10.1016/0025-5564(70)90132-X

[45] Tikhonov, A. N. On the Solution of Ill-Posed Problems and the Method of Regularization. Doklady Akademii Nauk SSSR 151 (1963), 501–504. English translation: Solution of Incorrectly Formulated Problems and the Regularization Method, Soviet Mathematics Doklady 4 (1963), 1035–1038.
https://www.mathnet.ru/eng/dan/v151/i3/p501

[46] Kuhn, H. W., and A. W. Tucker. Nonlinear Programming. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 1951, 481–492. The symposium was held in 1950.
https://projecteuclid.org/euclid.bsmsp/1200500249
https://digicoll.lib.berkeley.edu/record/112776

[47] Boyd, S., and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004. https://web.stanford.edu/~boyd/cvxbook/

[48] Chinneck, J. W., and E. W. Dravnieks. Locating Minimal Infeasible Constraint Sets in Linear Programs. ORSA Journal on Computing 3 (1991), 157–168. https://doi.org/10.1287/ijoc.3.2.157

[49] Nash, J. F. Equilibrium Points in n-Person Games. Proceedings of the National Academy of Sciences 36 (1950), 48–49. https://doi.org/10.1073/pnas.36.1.48

[50] Pearl, J. Causal Diagrams for Empirical Research. Biometrika 82 (1995), 669–688. https://doi.org/10.1093/biomet/82.4.669

[51] Scott, D., and C. Strachey. Toward a Mathematical Semantics for Computer Languages. Oxford Programming Research Group, PRG-6, 1971. https://www.cs.ox.ac.uk/monographs/cs/1971.html

[52] Milner, R. A Theory of Type Polymorphism in Programming. Journal of Computer and System Sciences 17 (1978), 348–375. https://doi.org/10.1016/0022-0000(78)90014-4

[53] Floyd, R. W. Assigning Meanings to Programs. Proceedings of Symposia in Applied Mathematics 19 (1967), 19–32. https://www.cse.chalmers.se/edu/year/2017/course/TDA384_LP1/files/lectures/additional-material/AssigningMeanings1967.pdf

[54] Hoare, C. A. R. An Axiomatic Basis for Computer Programming. Communications of the ACM 12 (1969), 576–580 and 583. https://doi.org/10.1145/363235.363259

[55] Cousot, P., and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. Proceedings of POPL 1977, 238–252. https://cs.nyu.edu/~pcousot/COUSOTpapers/POPL77.shtml

[56] Pnueli, A. The Temporal Logic of Programs. Proceedings of the 18th Annual Symposium on Foundations of Computer Science, 1977, 46–57. https://doi.org/10.1109/SFCS.1977.32

[57] Clarke, E. M., and E. A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. Logics of Programs, edited by D. Kozen, LNCS 131. Springer, 1982, 52–71; workshop held in 1981. https://doi.org/10.1007/BFb0025774

[58] Queille, J.-P., and J. Sifakis. Specification and Verification of Concurrent Systems in CESAR. International Symposium on Programming, LNCS 137. Springer, 1982, 337–351. https://doi.org/10.1007/3-540-11494-7_22

[59] Clarke, E., O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-Guided Abstraction Refinement. Computer Aided Verification, LNCS 1855. Springer, 2000, 154–169. https://doi.org/10.1007/10722167_15

[60] Davis, M., and H. Putnam. A Computing Procedure for Quantification Theory. Journal of the ACM 7 (1960), 201–215. https://doi.org/10.1145/321033.321034

[61] Robinson, J. A. A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM 12 (1965), 23–41. https://doi.org/10.1145/321250.321253

[62] Nieuwenhuis, R., A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T). Journal of the ACM 53 (2006), 937–977. https://doi.org/10.1145/1217856.1217859 https://www.cs.upc.edu/~roberto/papers.html

[63] Wetzler, N., M. J. H. Heule, and W. A. Hunt Jr. DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs. SAT 2014, LNCS 8561, 422–429. https://doi.org/10.1007/978-3-319-09284-3_31

[64] Heule, M. J. H., O. Kullmann, and V. W. Marek. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. SAT 2016, LNCS 9710, 228–245. Preprint: arXiv:1605.00723.
https://doi.org/10.1007/978-3-319-40970-2_15
https://arxiv.org/abs/1605.00723

[65] Reiter, R. A Theory of Diagnosis from First Principles. Artificial Intelligence 32 (1987), 57–95.
https://doi.org/10.1016/0004-3702(87)90062-2

[66] de Moura, L., and S. Ullrich. The Lean 4 Theorem Prover and Programming Language. Automated Deduction, CADE 28, LNCS 12699. Springer, 2021, 625–635.
https://doi.org/10.1007/978-3-030-79876-5_37

[67] The mathlib Community. The Lean Mathematical Library. Proceedings of CPP 2020, 367–381. Preprint first submitted in 2019.
https://doi.org/10.1145/3372885.3373824
https://arxiv.org/abs/1910.09336

[68] Kempe, A. B. On the Geographical Problem of the Four Colours. American Journal of Mathematics 2 (1879), 193–200.
https://www.jstor.org/stable/2369235

[69] Heawood, P. J. Map-Colour Theorem. Quarterly Journal of Pure and Applied Mathematics 24 (1890), 332–338.

[70] Appel, K., and W. Haken. Every Planar Map Is Four Colorable. Bulletin of the American Mathematical Society 82 (1976), 711–712. Announcement preceding the detailed 1977 papers.
https://doi.org/10.1090/S0002-9904-1976-14122-5

[71] Gonthier, G. Formal Proof—The Four-Color Theorem. Notices of the American Mathematical Society 55 (2008), 1382–1393.
https://www.ams.org/notices/200811/tx081101382p.pdf

[72] Hales, T., et al. A Formal Proof of the Kepler Conjecture. Forum of Mathematics, Pi 5 (2017), e2. Project completed in 2014; preprint appeared in 2015.
https://doi.org/10.1017/fmp.2017.1

[73] Gonthier, G., et al. A Machine-Checked Proof of the Odd Order Theorem. Interactive Theorem Proving, LNCS 7998. Springer, 2013, 163–179. Formalization completed in 2012.
https://doi.org/10.1007/978-3-642-39634-2_14

[74] Wilkinson, M. D., et al. The FAIR Guiding Principles for Scientific Data Management and Stewardship. Scientific Data 3 (2016), 160018.
https://doi.org/10.1038/sdata.2016.18

[75] Polu, S., and I. Sutskever. Generative Language Modeling for Automated Theorem Proving. arXiv:2009.03393, 2020.
https://arxiv.org/abs/2009.03393

[76] Yang, K., et al. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. Advances in Neural Information Processing Systems 36, 2023; arXiv:2306.15626.
https://arxiv.org/abs/2306.15626

[77] Romera-Paredes, B., et al. Mathematical Discoveries from Program Search with Large Language Models. Nature 625 (2024), 468–475. First published online in December 2023.
https://doi.org/10.1038/s41586-023-06924-6

[78] Trinh, T. H., et al. Solving Olympiad Geometry without Human Demonstrations. Nature 625 (2024), 476–482.
https://doi.org/10.1038/s41586-023-06747-5

[79] Eilenberg, S., and S. Mac Lane. General Theory of Natural Equivalences. Transactions of the American Mathematical Society 58 (1945), 231–294.
https://doi.org/10.1090/S0002-9947-1945-0013131-6

[80] Rutten, J. J. M. M. Universal Coalgebra: A Theory of Systems. Theoretical Computer Science 249 (2000), 3–80.
https://doi.org/10.1016/S0304-3975(00)00056-6
https://ir.cwi.nl/pub/48

[81] Vostokov, D. Theoretical Software Diagnostics: Collected Articles. Fourth edition. OpenTask, 2024. See “Five Golden Rules” for 4WS: What, When, Where, Why, and Supporting information.

[82] Vostokov, D. Trace, Log, Text, Narrative, Data: An Analysis Pattern Reference for Information Mining, Diagnostics, Anomaly Detection. Fifth edition. OpenTask, 2023; revision 5.06, August 2026. ISBN 978-1-912636-58-7.